PROGETTO FINALIZZATO
SISTEMI INFORMATICI E CALCOLO PARALLELO

SOTTOPROGETTO N. 5

Sistemi Evoluti per Basi di Dati

Coordinatore Prof. Domenico Saccà

Stefano Mizzaro*

La componente esperta del sistema FIRE

N. R/5/137 **Dicembre 1994**

Rapporto tecnico

***Dipartimento di Matematica e Informatica**
Università di Udine
33100 UDINE, ITALY

Sommario

Il sistema FIRE (*Flexible Information Retrieval Environment*) è un sistema di *information retrieval* in cui vengono usate tecniche di intelligenza artificiale, o più precisamente di rappresentazione della conoscenza, allo scopo di migliorare le prestazioni del sistema. Le parti di FIRE in cui tali tecniche vengono utilizzate ne costituiscono la cosiddetta componente esperta e sono concentrate in uno specifico modulo denominato IRES (*Information Retrieval Expert System*), descritto in questo documento. In tale modulo sono rappresentate alcune delle competenze e capacità che gli intermediari umani di sistemi di information retrieval adottano durante l'attività di ricerca di documenti. La descrizione di IRES è dapprima effettuata ad alto livello di astrazione e poi si spinge fino ad un livello di dettaglio molto vicino al codice.

Abstract

The FIRE (*Flexible Information Retrieval Environment*) system is an information retrieval system in which artificial intelligence, or more precisely knowledge representation, techniques are used in order to improve the system performance. The parts of FIRE in which such techniques are used form the so called expert component and they are concentrated in a specific module named IRES (*Information Retrieval Expert System*), described in this document. In such module some of the competence and capabilities that human intermediaries of information retrieval systems adopt during the document search activity are represented. IRES's description is first carried out at a high level of abstraction, and then it goes further until a level of detail very near to the code.

1. Introduzione

Il progetto FIRE (acronimo di *Flexible Information Retrieval Environment*) si situa nell'area dell'*Intelligent Information Retrieval* (IIR), ossia quel settore dell'*Information Retrieval* (IR) in cui si studiano le applicazioni di tecniche di *Intelligenza Artificiale* (IA) ai *Sistemi di IR* (IRS). I sistemi così ottenuti sono denominati *Sistemi di Information Retrieval Intelligenti* (IIRS).

Parte fondamentale del progetto FIRE è la realizzazione di un IIRS, il sistema FIRE. La componente esperta di tale sistema, ossia la parte del sistema in cui si utilizzano le tecniche d'IA, è concentrata in un preciso modulo, denominato IRES (*Information Retrieval Expert System*). IRES è un sistema esperto, il cui scopo è simulare alcune attività tipiche di un *intermediario* di un sistema d'IR tradizionale.

Questo documento ha lo scopo di descrivere l'implementazione della versione attuale di IRES. Più in dettaglio: la sezione 2 contiene alcuni richiami alla teoria dell'IR allo scopo di definire la terminologia adottata nel seguito. Nella sezione 3 è illustrato a grandi linee il funzionamento di FIRE, attraverso una breve descrizione dell'interfaccia utente e dell'architettura generale. Nelle sezioni 4, 5 e 6 è presentata la componente esperta del sistema; tale presentazione è organizzata ispirandosi ad alcune considerazioni riportate in Guida e Tasso (1994): dapprima IRES viene illustrato a livello *concettuale*, poi a livello *logico* e infine a livello *tecnico*, con alcuni riferimenti al codice. La sezione 7 conclude il lavoro.

2. Richiami di Information Retrieval

Prima di affrontare la descrizione del sistema FIRE, in questa sezione sono richiamati alcuni concetti base della disciplina dell'IR. La descrizione effettuata qui è ovviamente incompleta; per una trattazione più dettagliata si può fare riferimento a Salton (1989) e Ingwersen (1992).

2.1. L'IR tradizionale

I 5 attori principali dello scenario concettuale dell'IR sono l'autore (o gli autori), l'indicizzatore (gli indicizzatori), l'IRS, l'utente e l'intermediario. Tale scenario è schematizzato in figura 1 e illustrato nel seguito.

L'*autore* è colui che, per qualche motivo, memorizza le proprie conoscenze, nozioni in un *documento* (un libro, un articolo, ecc.). Tale documento può venire memorizzato in una cosiddetta *banca dati*; Le informazioni gestite dalle banche dati possono essere di varia natura; un caso comune è quello delle *banche dati bibliografiche*, le quali contengono riferimenti bibliografici a documenti.

Per facilitarne il successivo reperimento, il documento memorizzato in una banca dati viene sottoposto ad un'operazione di *indicizzazione*, effettuata dal secondo attore dello scenario: l'*indicizzatore*. Con tale operazione, al documento vengono associate alcune *parole chiave*, da usarsi durante le operazioni di reperimento.

L'*IRS* fornisce le funzioni che permettono di gestire una o più banche dati: deve permettere la memorizzazione delle informazioni e l'accesso ad

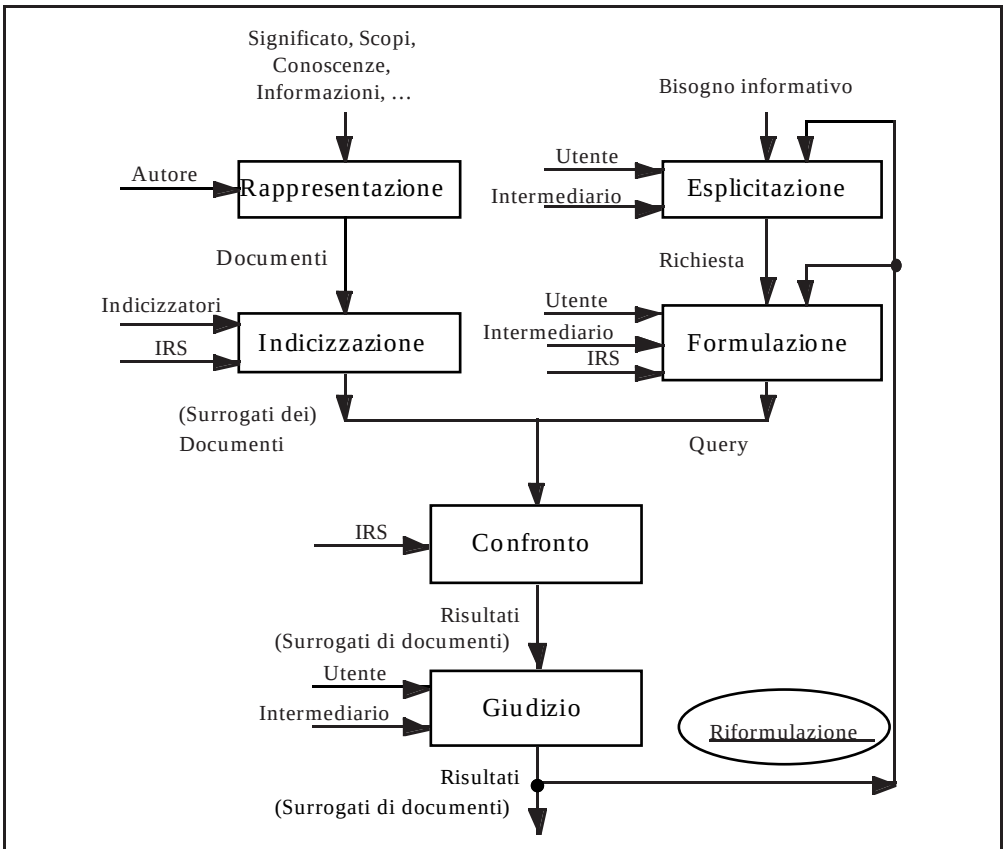

Figura 1. Lo scenario concettuale dell'IR.

esse.

L'*utente* si presenta ad un IRS con un *bisogno informativo* (BI) da soddisfare. La situazione in cui si trova un utente di un IRS è stata descritta in modo esauriente da un punto di vista cognitivo in Belkin (1980): l'utente si trova in uno stato di conoscenza *anomalo* (o *incompleto*, secondo Ingwersen, 1992) e ha necessità di informazioni che gli permettano di colmare una sua lacuna conoscitiva. Il BI di un utente risulta quindi essere il bisogno di quelle informazioni che gli consentirebbero di uscire dal suo stato anomalo o incompleto di conoscenza.

Il BI è a livello inconscio; la sua esplicitazione, solitamente effettuata dall'utente in linguaggio naturale, è denominata *richiesta*. L'attività di esplicitazione del BI è spesso più complessa di quanto sembri a prima vista, in quanto l'utente non sempre riesce ad esprimersi correttamente, proprio a causa del fatto che il BI riguarda un settore di cui egli non ha una buona conoscenza; inoltre, egli può essere influenzato negativamente da fattori esterni.

Un primo compito dell'*intermediario*, che funge da mediatore fra utente e IRS, è di aiutare l'utente a esplicitare il più correttamente e completamente possibile il suo BI.

Per poter essere compresa dall'IRS, la richiesta deve essere espressa in un opportuno linguaggio formale: tale espressione della richiesta (e del BI) è denominata *query*. La traduzione da richiesta a query è solitamente effettuata dall'intermediario, il quale funge anche da tramite fra utente e sistema al fine di evitare le difficoltà tecniche connesse all'uso di quest'ultimo. La query viene poi sottoposta all'IRS, il quale risponde restituendo le informazioni che la soddisfano; queste informazioni possono essere documenti o, nel caso di banche dati bibliografiche, *surrogati* di tali documenti, ovvero una loro breve descrizione unita alla referenza bibliografica.

Spesso il primo tentativo di formulazione del BI in una forma comprensibile dal sistema non dà i risultati desiderati: si rende quindi necessaria una fase di ri-espressione del BI, o di *riformulazione della query* (si veda Meadow and Cochrane, 1981), che può essere anche iterata più volte fino al raggiungimento dell'obiettivo prefissato (la raccolta delle informazioni necessarie a soddisfare il BI). L'intermediario interviene anche in questo processo, in quanto la fase di riformulazione avviene sotto il suo controllo. Un'ipotesi di lavoro accettata in FIRE è che la riformulazione della query abbia anche un altro effetto, ossia aiuti l'utente a comprendere il proprio BI e quindi ad esprimerlo in modo migliore. L'attività di comprensione ed esplicitazione del BI non è quindi da ritenersi terminata una volta effettuata la formulazione della prima richiesta: essa continua durante tutto il processo di riformulazione della query.

Essendo lo scopo di IRES la simulazione di alcune delle attività dell'intermediario umano, è opportuno analizzare più in dettaglio tali attività. Riassumendo quanto detto poc'anzi a tale proposito, si può dire che l'intermediario abbia quattro compiti principali: aiutare l'utente ad esprimere il BI, tradurre la richiesta in query, guidare l'utente durante la fase di riformulazione della query e isolare l'utente dalle difficoltà tecniche di utilizzo di un IRS. Solitamente, l'intermediario ha a sua disposizione alcune fonti di conoscenza che gli permettono di eseguire tali attività:

- un *thesaurus* (si veda Danesi, 1990), contenente conoscenza terminologica su un dato dominio;
- sé stesso, in quanto usualmente conosce almeno a grandi linee il contenuto delle banche dati, si fa un'idea dell'utente con cui interagisce, sa organizzare l'intero processo di interazione fra utente e IRS e possiede altre conoscenze e capacità; tutto ciò costituisce la cosiddetta *conoscenza esperta*, ovvero un insieme di tecniche che gli consentono di svolgere nel modo migliore le attività di supporto all'utente sopra elencate;
- l'utente, che possiede un BI e talvolta (raramente) anche una buona conoscenza dell'argomento su cui viene effettuata la ricerca.

Come risulterà chiaro più avanti, la riformulazione è l'attività principale di IRES. In considerazione di ciò è importante rilevare che non esiste uno standard capace di strutturare in maniera razionale l'attività di riformulazione. L'intermediario umano, inoltre, presenta alcune limitazioni intrinseche nella propria natura umana; ciò rende indubbiamente interessante il tentativo di affidare al calcolatore questo compito.

2.2. L'IR intelligente

Vi sono alcune similitudini fra il campo dell'IR e quello delle basi di dati: in entrambi i casi si parla di rappresentazione delle informazioni, di query, di processo di reperimento. Vi sono però anche importanti differenze, che separano nettamente i due campi: nel caso delle basi di dati, le informazioni sono completamente strutturate, e questa caratteristica è pesantemente utilizzata nel processo di reperimento, mentre nelle informazioni gestite dagli IRS la strutturazione è solamente parziale; inoltre, nel caso dell'IR vi è il problema dell'espressione del BI.

Queste differenze rendono il problema dell'IR un problema *difficile*: i sistemi di gestione delle basi di dati reperiscono *dati*, mentre gli IRS reperiscono *informazioni*. Si può dire che i sistemi di gestione delle basi di dati lavorino a livello sintattico, gli IRS a livello semantico.

Come conseguenza di ciò, si ha che le prestazioni di un IRS (misurabili in termini di *recall* e *precision*, si veda ad esempio Salton, 1989) non sono ottimali. Vi sono dunque dei margini di miglioramento; una strada percorribile per ottenere risultati migliori è l'IIR, ossia l'utilizzo di tecniche d'IA nella costruzione di un IRS. Le tecniche d'IA sembrano indispensabili per conseguire risultati ottimali, come sostenuto ad esempio in Croft (1993).

In un IIRS, l'utente esprime il suo BI utilizzando:

- i termini che gli sembrano più adatti;
- il numero di documenti desiderati (*range*);
- l'obiettivo della ricerca (*High-recall*, *High-precision*);
- altre informazioni utilizzabili dal sistema, quali relazioni semantiche fra concetti, scopo della ricerca, ecc.

Un IIRS interagisce direttamente con l'utente. Esso deve quindi simulare il comportamento dell'intermediario: aiutare l'utente ad esprimere il suo BI in modo completo, permettergli di effettuare l'espressione del BI in maniera semplice e supportarlo nell'attività di riformulazione. Inoltre, un IIRS deve essere dotato di un'interfaccia utente usabile, in quanto l'utente che interagisce direttamente con il sistema non deve essere ostacolato da dettagli tecnici che ne rendano difficoltoso l'uso.

In un IIRS le fonti di conoscenza elencate poc'anzi che un intermediario utilizza vengono implementate da basi di conoscenza del sistema. Si può pensare ad esempio alle seguenti basi di conoscenza (Brooks, 1987):

- conoscenza esperta: modella la conoscenza esperta, ossia le tecniche che un intermediario usa durante l'interazione con l'utente. In questa base di conoscenza saranno contenute ad esempio conoscenze per pianificare la fase di riformulazione e conoscenze di tipo linguistico per poter comprendere la richiesta (o almeno per poterla tradurre in query);
- conoscenza terminologica del dominio: è costituita da un thesaurus, solitamente arricchito con altre informazioni, che consente al sistema di scegliere i termini corretti nelle fasi di espressione del BI e di riformulazione della query;
- conoscenze morfologiche, ossia algoritmi di troncamento utilizzati per inserire nella query termini troncati o varianti morfologiche dei termini già presenti;
- conoscenza sugli utenti: modella le conoscenze sulla popolazione di utenti che accedono al sistema (stereotipi) e sui singoli utenti (modelli di specifici utenti);
- conoscenza della banca dati: contiene conoscenze (solitamente ad un livello molto generale ed incompleto) sul contenuto di una banca dati, sugli argomenti trattati nei documenti di una banca dati, sulle modalità di accesso, e così via.

Tali basi di conoscenza possono essere utilizzate dall'IIRS per comprendere il BI (attività che comunque non è indispensabile, in quanto talvolta neppure l'intermediario umano lo fa) e/o per riformulare la query: la base di conoscenza esperta contiene le conoscenze necessarie per scegliere quali termini proporre all'utente fra tutti i termini ricavabili navigando sul thesaurus o utilizzando tecniche morfologiche (per produrre troncamenti o termini simili morfologicamente).

Questo termina la breve descrizione del campo dell'IR. Nelle prossime sezioni è presentato il sistema FIRE.

3. Il sistema FIRE

Il sistema FIRE è un IIRS per banche dati bibliografiche ed è descritto in Floreanini (1993) e in Brajnik et al. (1990a, 1990b). Qui si evita di presentarlo in modo completo: il funzionamento è illustrato unicamente in linea generale.

In FIRE si perseguono diversi obiettivi. Lo si vuole utilizzare come piattaforma sperimentale per studiare le modalità di sviluppo di interfacce intelligenti a IRS (per banche dati bibliografiche), quindi deve essere modulare e modificabile. Inoltre, FIRE deve essere in grado di isolare l'utente dalle difficoltà tecniche di utilizzo degli IRS, grazie ad un'interfaccia utente *user-friendly*.

Il suo scopo principale è comunque quello di simulare alcune funzionalità tipiche di un intermediario, supportando l'utente non solo da un punto di vista tecnico, ma anche concettuale. Uno dei compiti di FIRE è ad esempio guidare l'utente durante la fase di riformulazione della query.

Il processo di comprensione del BI da parte dell'utente non viene supportato in maniera esplicita dal sistema, ma, come detto, la speranza è che durante la fase di riformulazione l'attività del sistema consenta all'utente di arrichire la propria conoscenza del settore in cui necessita di informazioni, e quindi riesca a comprendere ed esprimere in modo migliore le proprie esigenze.

Allo stato attuale, FIRE non è invece in grado di sostituire completamente l'intermediario, in quanto non supporta (se non da un punto di vista tecnico, tramite l'interfaccia *user-friendly*) l'utente nell'attività di traduzione della richiesta in query.

Per descrivere in linea generale il sistema FIRE, nel paragrafo 3.1 è illustrato il funzionamento dal punto di vista dell'utente, ponendo l'accento sull'interfaccia utente; nel paragrafo 3.2 è presentata l'architettura generale del sistema.

3.1. Interfaccia utente di FIRE

Allo scopo di comprendere il funzionamento di FIRE, ne è qui illustrata brevemente l'interfaccia utente, progettata sulla base delle specifiche in Brajnik et al. (1991b).

Come detto, il sistema FIRE non permette all'utente di esprimere una richiesta in linguaggio naturale, ma accetta una query in forma booleana. È quindi compito dell'utente individuare i concetti che esprimono nel modo migliore (almeno in prima approssimazione) il proprio BI, individuare i termini con cui tali concetti possono essere espressi e raggruppare opportunamente tali termini in *faccette* (i termini in una stessa faccetta sono in *or* logico e le faccette sono in *and* fra di loro).

Unitamente alla query booleana, l'utente può specificare anche altre caratteristiche del proprio BI, i cosiddetti *vincoli sulla ricerca.* Nella versione attuale di FIRE tali vincoli si limitano all'obiettivo della ricerca (alto richiamo o alta precisione) e al numero di documenti desiderati, espresso tramite un intervallo di interi.

L'aspetto dell'interfaccia è illustrato in figura 2. A sinistra vi è il `Query Panel`, in cui l'utente può specificare le faccette e i termini che descrivono il proprio BI e i vincoli sulla ricerca; ad ogni termine l'utente può associare un grado di interesse (alto o basso), che indica quanto tale termine sia importante per il suo BI. A destra vi è un altro pannello in cui vengono visualizzati i titoli dei documenti reperiti (documenti di cui l'utente può visualizzare il surrogato cliccando sul titolo prescelto). L'interfaccia fornisce anche cinque bottoni:

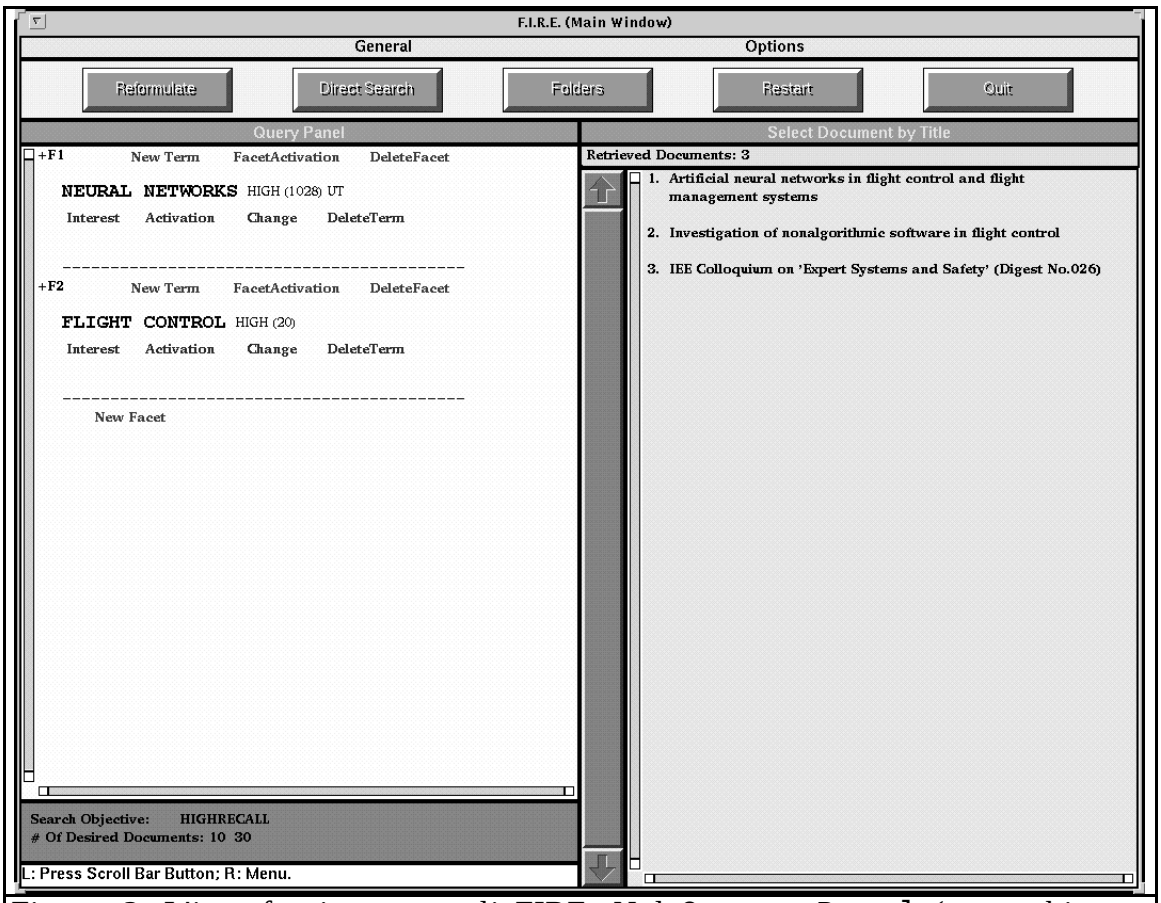

Figura 2. L'interfaccia utente di FIRE. Nel `Query Panel` (zona chiara a sinistra) è specificata la query "Neural networks and flight control"; a destra, nella zona grigia, sono visualizzati i titoli dei documenti corrispondenti e in alto sono situati i cinque bottoni e i due menu a disposizione dell'utente.

- `Direct Search`, che reperisce i (titoli dei) documenti che soddisfano la query;
- `Reformulate`, che avvia l'attività di riformulazione;
- `Folders`, che consente all'utente di catalogare i documenti reperiti, suddividendoli in rilevanti, non rilevanti, utili, non utili, ecc.;
- `Restart`, che consente di inizializzare il sistema;
- `Quit`, per uscire da FIRE.

Vi sono infine due menu, tramite i quali si può ad esempio scegliere la banca dati su cui lavorare (ne sono attualmente disponibili quattro). L'interfaccia mette a disposizione anche altre funzionalità, che non interessano qui; per maggiori dettagli si veda Floreanini (1993).

L'interazione FIRE-utente può avvenire in due modalità, denominate elaborazione *manuale* e *semiautomatica*. Nel primo caso, l'utente utilizza FIRE come un normale sistema booleano (sfruttando le potenzialità fornite dall'interfaccia a finestre): esprime il proprio BI, effettua una ricerca, esamina i titoli dei documenti reperiti, ne visualizza i surrogati. Se i risultati sono soddisfacenti, l'interazione con FIRE termina, altrimenti l'utente deve modificare la query iniziale. Ad esempio, se la ricerca precedente ha reperito pochi documenti (come accade solitamente con le

attuali versioni di FIRE e delle banche dati) l'utente può espandere la query aggiungendo termini sinonimi a quelli utilizzati in precedenza. Per individuare tali sinonimi, l'utente può, oltre che riflettere più a fondo sul proprio BI, consultare le descrizioni dei documenti reperiti, o utilizzare un thesaurus. Una volta modificata la query, il procedimento viene iterato: si effettua una nuova ricerca, si valuta l'insieme di documenti reperiti dal sistema e così via.

Le potenzialità di FIRE vengono però sfruttate appieno con la modalità di interazione semiautomatica, innescata dal bottone `Reformulate`: FIRE guida il processo di riformulazione, suggerendo le modifiche da effettuare alla query. È importante notare il carattere *supportivo* di FIRE: il sistema non modifica autonomamente la query, ma si limita a suggerire tali modifiche all'utente, il quale accetta o meno le proposte del sistema. Esemplificando nel caso dell'espansione della query, il sistema propone i termini da aggiungere come sinonimi nelle faccette; l'utente deve indicare quali dei termini proposti migliorano effettivamente l'espressione del suo BI.

Per poter svolgere l'attività di supporto all'utente, FIRE contiene le basi di conoscenza seguenti, che modellano (alcune del) le conoscenze utilizzate da un intermediario (si veda la sezione 2):

- DSKB (*Domain Specific Knowledge Base*): è costituita da un thesaurus, arricchito con alcune informazioni, quali il *posting count* dei termini, il loro *livello di specificità* (si veda Brajnik et al. 1991a) e un flag che indichi se i termini sono controllati o non controllati. Nel seguito, con 'thesaurus' mi riferirò a tale base di conoscenza;
- EKB (*Expert Knowledge Base*): questa base di conoscenza modella (parte del) la conoscenza esperta. È costituita da conoscenze morfologiche (algoritmi che individuano la radice di una parola) e da conoscenze sull'attività di riformulazione, espresse in parte algoritmicamente e in parte sottoforma di *regole di produzione* (si veda ad esempio Rich, 1986). Sarà analizzata più in dettaglio nel seguito.

3.2. Architettura generale di FIRE

Per completare la descrizione a livello generale di FIRE, è qui presentata l'architettura del sistema, illustrata in figura 3. In tale figura compaiono le seguenti componenti:

- Il modulo `UI` (User Interface), interfaccia utente realizzata secondo il paradigma WIMP (Window Icon Menu Pointer). L'interfaccia può essere scomposta in due sottomoduli: `IRESFACE`, che costituisce il principale canale di comunicazione con l'utente e il `DSKBED` (acronimo per DSKB EDitor) utilizzabile per accedere al thesaurus, sia in modalità di sola lettura (utile durante la fase di riformulazione della query), sia per aggiungere nuovi termini, nuove relazioni, ecc. Il modulo `IRESFACE` è stato brevemente illustrato poc'anzi.
- Il modulo `IRES` (Information Retrieval Expert Subsystem), incaricato della simulazione dell'intermediario. È un *sistema esperto* (Rich, 1986)

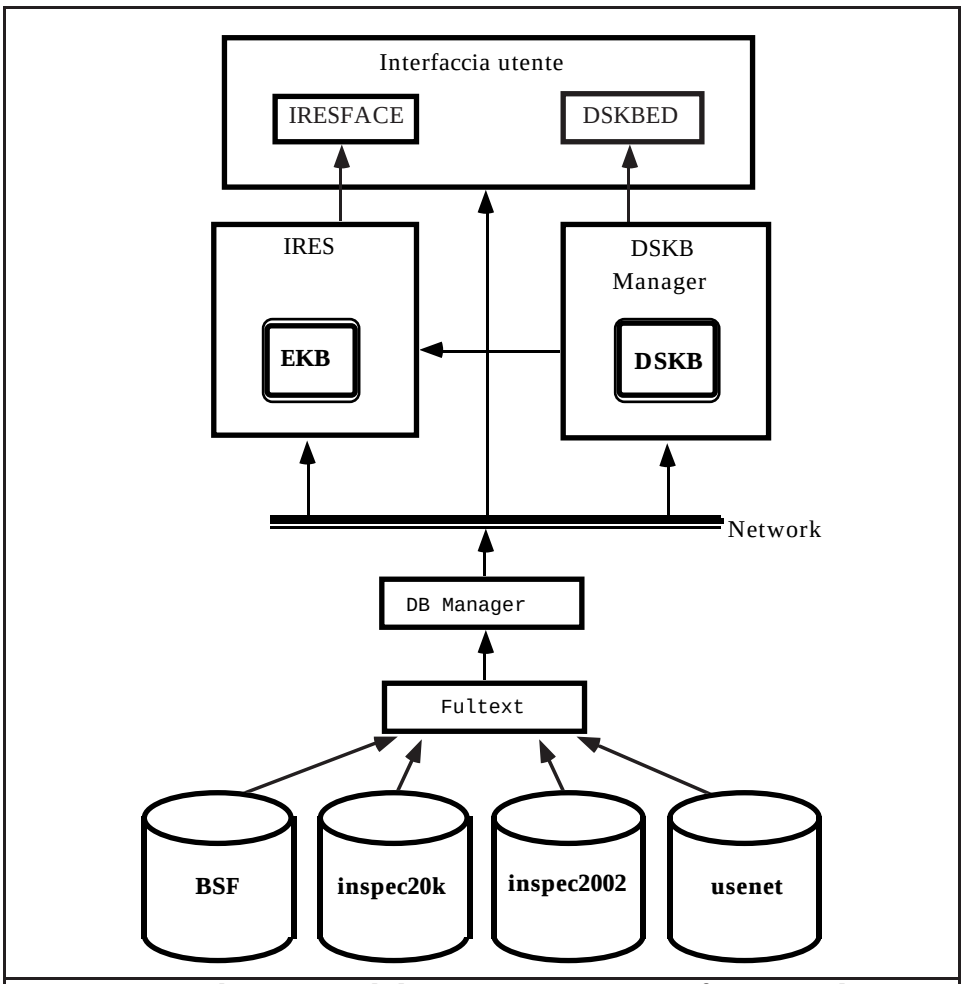

Figura 3. Architettura del sistema FIRE. Le frecce indicano i servizi forniti da alcune componenti ad altre componenti.

e contiene la base di conoscenza EKB, che modella le conoscenze di un intermediario. Questo modulo sarà descritto dettagliatamente nelle prossime sezioni.

- Il modulo DSKBMAN (da DSKB MANager), incaricato della gestione della DSKB. L'utente può accedere ai servizi del DSKBMAN tramite l'interfaccia DSKBED che, come detto, permette di esaminare e modificare il thesaurus.
- Il modulo DBMAN (Data Base MANager), gestore delle banche dati a cui il sistema accede. Il modulo DBMAN fornisce agli altri moduli i servizi necessari per l'accesso alle banche dati, utilizzando le funzioni dell'IRS FulText.
- L'IRS booleano FulText commercializzato da Fulcrum S.p.A. È un IRS commerciale di cui si sfrutta la capacità di accedere in modo efficiente alle banche dati.
- Le 4 banche dati bibliografiche. Per ora FIRE può accedere a 4 banche dati (o *collezioni*). La prima a sinistra in figura è denominata BSF (acronimo di Bibliografia Storica Friulana) ed è costituita da circa 5000 documenti relativi ad aspetti storici, economici e legislativi della vita montana del Friuli-Venezia Giulia. Procedendo verso destra, vi sono due collezioni ricavate dalla banca dati INSPEC e relative alle applicazioni di tecniche di intelligenza artificiale nell'industria; la

prima di esse contiene 20.000 documenti, la seconda 2000. L'ultima collezione, indicata in figura con l'etichetta 'usenet', è ricavata raccogliendo circa 5000 *posting* su Usenet, da differenti gruppi di interesse.

L'intero sistema è implementato su una rete di SUN SparcStations, con sistema operativo Solaris. Nell'implementazione di FIRE si sono utilizzati diversi linguaggi di programmazione: il DBMAN è implementato in C, l'EKB in XPSE, un sistema a regole di produzione sviluppato presso il Dipartimento di Matematica e Informatica dell'Università di Udine e il resto del sistema in Lucid Common LISP e CLOS, ad eccezione delle interfacce grafiche (IRESFACE e DSKBED), per le quali si è utilizzato il CLIM.

In questo documento non sono illustrati ulteriori dettagli del sistema FIRE. Le prossime tre sezioni sono dedicate alla descrizione del modulo IRES., descritto a livelli di dettaglio sempre maggiori: dapprima a livello concettuale, poi a livello logico e infine a livello tecnico (si veda Guida e Tasso, 1994).

4. Il modulo IRES: descrizione concettuale

In questa sezione è descritto il modulo IRES ad un livello *concettuale* (Guida e Tasso, 1994): i dettagli relativi all'implementazione verranno presentati nelle prossime sezioni. Sono illustrati i concetti di riformulazione, direzione di riformulazione, tattiche, piani, focus, candidati, ordinamento dei candidati e viene sinteticamente descritto il flusso del controllo in IRES.

4.1. Riformulazione e direzione di riformulazione

La *riformulazione* è l'attività svolta congiuntamente da sistema e utente al fine di meglio definire e comprendere il BI espresso in precedenza dall'utente (si veda la figura 1). Il compito principale del sistema è la proposta all'utente di opportune modifiche da apportare alla query al fine di esprimere in modo più preciso e completo il BI; l'utente deve scegliere fra tali modifiche quelle secondo lui più appropriate. La riformulazione è quindi una sequenza di *operazioni di modifica della query* (OMQ), proposte dal sistema e confermate o meno dall'utente.

Un altro concetto importante per comprendere l'attività di IRES è quello di *direzione di riformulazione* (DR). Si ricordi che l'utente, oltre ad indicare i termini che caratterizzano il proprio BI, specifica anche un intervallo indicante il numero di documenti desiderato ([lb,hb]); ora, la DR è definita in base a tale intervallo e al numero di documenti reperito dall'ultima ricerca (RetrDocs). Se RetrDocs < lb, IRES deve cercare di reperire un maggior numero di documenti, e in questo caso si dice che la DR è 'espandere' (o *expand*). Se al contrario RetrDocs > hb, la DR è 'restringere' (o *narrow*) e IRES cerca di modificare la query in modo da reperire un numero di documenti inferiore. Durante la riformulazione, il sistema effettua delle ricerche al fine di decidere se arrestare la riformulazione: se la DR è expand, la riformulazione viene sospesa se

`RetrDocs` ≥ `lb`; simmetricamente, se la DR è `narrow`, la riformulazione viene sospesa se `RetrDocs` ≤ `hb`.

A seconda della DR, la riformulazione deve quindi procedere in modi differenti, con OMQ diverse nei due casi. Se si deve espandere, si può ad esempio aggiungere termini sinonimi in or logico con altri termini in una faccetta, o eliminare una o più faccette. Se si deve restringere, due fra le OMQ possibili sono l'eliminazione di termini, l'introduzione di termini in or in faccette *negative* (ossia in faccette contenenti termini che non devono essere presenti nei documenti reperiti), l'aggiunta di faccette o l'eliminazione di *campi di ricerca*[1] di alcuni termini.

In FIRE, le OMQ effettivamente implementate sono unicamente quelle che lavorano sui termini e possono essere suddivise in tre gruppi: quelle che aggiungono nuovi termini nella query, quelle che ne eliminano e quelle che sostituiscono un termine con un altro.[2] In realtà, in FIRE non si eliminano definitivamente i termini, ma ci si limita a *disattivarli*: un termine disattivato rimane visualizzato (con un aspetto grafico differente) nel `Query Panel`, in modo che l'utente possa in seguito riattivarlo, ma tale termine non ha nessuna influenza nelle ricerche sulla banca dati.

Le OMQ che aggiungono termini necessitano di un modo per reperire o generare un insieme di termini alternativi a quelli presenti nella query e di un criterio per scegliere fra di essi; le OMQ che disattivano termini necessitano di criteri per scegliere quali termini disattivare fra i termini della query. Le OMQ che sostituiscono termini necessitano di entrambi, in quanto una sostituzione può essere vista come una disattivazione seguita da un'aggiunta.

Per reperire i nuovi termini con cui modificare la query `IRES` ha a disposizione la DSKB e le tecniche morfologiche. Se alcuni dei termini della query compaiono nella DSKB, *navigando* su di essa (ovvero reperendo i termini in relazione con i termini di partenza) si possono infatti raggiungere termini correlati da relazioni semantiche con i termini presenti nella query. Le conoscenze morfologiche permettono di individuare termini presenti nella DSKB e simili morfologicamente (termini con la stessa radice) a quelli che occorrono nella query ovvero i *troncamenti* di essi.

Nel caso che nella query iniziale compaiano termini non appartenenti al thesaurus, il sistema cerca di trovare termini correlati ad essi nel thesaurus, possibilmente controllati; questo perché la riformulazione, essendo basata sulla navigazione nella DSKB, ha ovviamente maggiore efficacia se i termini nella query sono termini controllati del thesaurus. Questa operazione viene effettuata tramite analisi morfologica (reperimento nel thesaurus dei termini simili morfologicamente ad un termine dato) o sfruttando il *lead-in vocabulary* (ossia le relazioni UT della DSKB).

---

1 I campi di ricerca di un termine individuano le parti dei surrogati dei documenti in cui andare ad effettuare la ricerca di quel termine. Esempi di campi di ricerca sono: autore, titolo, abstract, tipo di documento, ecc.

2 Sono previste, ma non ancora implementate, anche la cancellazione di faccette, la fusione di più faccette (trasformazione di un and logico in un or), la suddivisione di una faccetta (trasformazione di un or logico in un and) e l'aggiunta e l'eliminazione dei campi di ricerca di un termine.

In generale, è possibile scegliere in ogni momento fra più OMQ alternative; restano quindi da illustrare le OMQ in modo dettagliato e i criteri implementati in IRES per scegliere in modo sensato quale fra di esse applicare di volta in volta. Questi argomenti sono affrontati nei prossimi paragrafi.

4.2. Tattiche e piani

Per descrivere le OMQ che IRES propone all'utente è necessario introdurre i concetti di tattica e piano.

Una *tattica* è un'OMQ *atomica*, ossia non ulteriormente scomponibile, per IRES: esso non e' in grado di scomporre ulteriormente le OMQ che propone all'utente. Le tattiche implementate in IRES sono derivate da quelle illustrate in Bates (1979) e in Brajnik et al. (1991a); esempi di tattiche sono: aggiungere in una faccetta i termini 'RT' (o 'BT' o 'NT') di un termine già appartenente a tale faccetta, aggiungere in una faccetta il termine troncato di un termine già appartenente a tale faccetta, disattivare un termine, eccetera.

Per quanto concerne la scelta delle OMQ da proporre all'utente, IRES non lavora a livello di tattiche, ma di sequenze di tattiche: le tattiche sono raggruppate in sequenze cablate denominate piani. Quindi IRES sceglie quale piano eseguire sulla base di alcuni criteri (che saranno illustrati nel seguito), dopodichè l'effettiva esecuzione del piano corrisponde all'esecuzione in sequenza delle tattiche che lo compongono.

Il raggruppamento delle tattiche in piani è giustificato da due ordini di motivi. Da un punto di vista concettuale, non sempre esistono criteri per poter scegliere in modo sensato fra due tattiche alternative, oppure può essere sensato far seguire in ogni caso l'applicazione di una tattica ad un'altra. Da un punto di vista pragmatico (secondo motivo), l'utilizzo dei piani permette di ottenere una maggiore efficienza temporale da parte del sistema.

Due esempi di piani sono: la successione delle 3 tattiche che propongono all'utente di aggiungere in una faccetta i termini, rispettivamente, morfologicamente simili, 'RT' e troncati di un termine di partenza già appartenente alla faccetta; la successione delle 2 tattiche che propongono di aggiungere in una faccetta i termini, rispettivamente, morfologicamente simili e 'RT' di un termine di partenza già appartenente alla faccetta. Il primo piano sarà tipicamente utilizzato in una situazione in cui l'obiettivo è 'alto richiamo', il secondo in una ricerca ad 'alta precisione', in quanto non effettua troncamenti, operazioni che aumentano il richiamo ma peggiorano la precisione.

La navigazione sulla DSKB avviene quindi in modo predeterminato, attraverso l'uso di tattiche e piani. Un elenco completo delle tattiche e dei piani implementati in IRES sarà riportato nel seguito, nella sezione 5.2.

4.3. Focus, candidati e ordinamento dei candidati

Sono qui illustrati i metodi adottati da IRES per scegliere quale piano proporre all'utente fra tutti quelli teoricamente proponibili.

Si osservi che un piano (e una tattica) vanno applicati ad un termine della query: si naviga sul thesauro *a partire da un termine della query*, si tronca *un termine della query*, si disattiva *un termine della query*, ecc.

In IRES, il *focus* è il termine da cui partire per eseguire un'operazione di modifica della query (ossia un piano). Terminata l'esecuzione di tale piano, il focus andrà ricalcolato, in quanto la query potrebbe essere modificata. La definizione di focus è derivata dall'omonimo concetto in Brajnik et al. (1991a): in tale lavoro, il focus è la faccetta su cui eseguire un'operazione di modifica della query (ossia, in quel caso, una tattica).

Dovrebbe risultare chiaro a questo punto che le possibili operazioni di modifica alla query sono individuate univocamente dal piano da applicare e dal focus a cui applicarlo. L'individuazione del focus è ovviamente importante quanto la scelta del piano da applicare: la coppia <focus,piano> è denominata *candidato*.

Ora, per scegliere fra le possibili operazioni di modifica della query quale effettuare, è sufficiente ordinare i candidati secondo un ordine (che in generale sarà parziale) di preferenza e scegliere uno dei massimali rispetto a tale ordinamento: se l'ordinamento è effettuato secondo criteri corretti, ai candidati massimali corrispondono le operazioni di modifica della query preferibili. I criteri a cui si è appena fatto riferimento saranno ovviamente basati sulle caratteristiche del candidato: tipo del piano (alcuni piani saranno, in certe situazioni, preferibili ad altri), caratteristiche del termine (ad esempio, il suo posting count – ossia il numero di documenti che contengono quel termine), caratteristiche della faccetta a cui il termine appartiene (ad esempio, il posting count della faccetta) e così via. Esemplificando, se si deve espandere una query con due faccette ognuna contenente un unico termine, è preferibile aggiungere sinonimi inizialmente nella faccetta con il posting count minore fra le due, in quanto è il 'collo di bottiglia' della situazione. Tali criteri costituiscono, come si vedrà nelle prossime sezioni, una parte della EKB di IRES.

4.4. Flusso di controllo in IRES

Per terminare la descrizione a livello logico di IRES ne è qui schematizzato il flusso di controllo, ossia la successione delle operazioni effettuate in IRES durante la riformulazione.

IRES opera ciclicamente: individua i candidati ammissibili, ne sceglie un massimale ed esegue l'OMQ corrispondente, ossia il piano del candidato. Dopodichè, IRES dovrà calcolare un nuovo insieme di candidati (essendo la query modificata, questo sarà differente dall'insieme di candidati del passo precedente), ri-individuare i massimali, ecc., ripetendo finché non si verifichino le condizioni per terminare il processo di riformulazione.

Tali condizioni, nell'IRES attuale, sono essenzialmente di due tipi: o il numero di documenti reperiti da una query rientra nell'intervallo di documenti desiderati ($lb \geq RetrDocs \geq hb$) o il sistema ha esaurito tutte le alternative e non vi sono più operazioni di modifica della query possibili. Il primo di tali criteri implica che IRES, durante la riformulazione, esegua delle ricerche; anche il sapere quale è il momento adatto per effettuare una ricerca è quindi un'informazione contenuta nelle basi di conoscenza di IRES.

L'utente ha comunque la possibilità di interrompere il process di riformulazione da interfaccia.

5. Il modulo IRES: descrizione logica

In questa sezione è descritto il modulo IRES ad un livello logico (Guida e Tasso, 1994): si evita quindi di fare esplicito riferimento al codice mantenendosi ad un livello più astratto. Vengono illustrate le basi di conoscenza utilizzate per rappresentare le conoscenze descritte nella sezione precedente. Nel paragrafo 5.1. è presentata la struttura generale delle basi di conoscenza di IRES; nei paragrafi 5.2 e 5.3 sono illustrate rispettivamente le tattiche ed i piani implementati e i criteri per la scelta dell'ordine di esecuzione delle operazioni.

5.1. Struttura delle basi di conoscenza di IRES

Come detto, IRES è un sistema esperto che modella in basi di conoscenza le conoscenze normalmente utilizzate da un intermediario al fine di simularne l'attività durante la fase di riformulazione. Tale simulazione consiste principalmente nel proporre nuovi termini alternativi o aggiuntivi a quelli già presenti nella query.

Si è detto in precedenza che le basi di conoscenza di IRES sono la DSKB che contiene conoscenza terminologica e l'EKB, che contiene conoscenze relative alla navigazione sulla DSKB (ossia tattiche e piani e criteri per la scelta dei candidati migliori). In realtà, la struttura delle basi di conoscenza di IRES può essere schematizzata in modo migliore: si veda la figura 4. In tale figura si vede come l'EKB sia composta dai criteri per la scelta dei candidati e da tattiche e piani. La definizione di tattiche e piani ha però bisogno di conoscenze di tipo terminologico, fornite dalla DSKB, e di tipo morfologico, fornite da algoritmi di troncamento. In FIRE vi sono 2 di tali algoritmi: per la lingua inglese si è implementato l'algoritmo di Porter (1980) e per la lingua italiana un più semplice algoritmo basato su un elenco di suffissi.

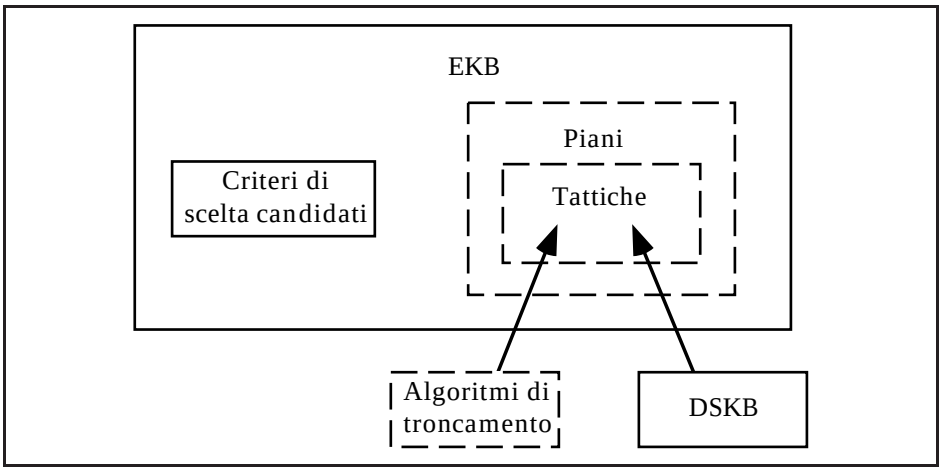

Figura 4. Struttura delle basi di conoscenza di FIRE. I rettangoli con i bordi tratteggiati indicano conoscenze di tipo algoritmico.

5.2. Tattiche e piani

È qui riportato l'elenco completo delle tattiche implementate in IRES, unitamente al loro effetto. Le tattiche sono raggruppate in 3 gruppi: quelle che aggiungono termini, quelle che ne sostituiscono e quelle che ne disattivano. Con <nome-tattica>(t) si intende che la tattica va applicata al termine t.

Tattiche che aggiungono termini:

- parallel-ut(t): propone all'utente una lista di termini da aggiungere nella faccetta in cui compare t e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono i termini legati a t da una relazione di tipo UT nel thesaurus. È importante in quanto può essere usata per aggiungere termini controllati a termini non controllati che compaiono nella query;
- parallel-rt(t): propone all'utente una lista di termini da aggiungere nella faccetta in cui compare t e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono i termini legati a t da una relazione di tipo RT nel thesaurus;
- super-add(t): propone all'utente una lista di termini da aggiungere nella faccetta in cui compare t e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono i termini legati a t da una relazione di tipo BT nel thesaurus, quindi termini più generali;
- sub-add(t): propone all'utente una lista di termini da aggiungere nella faccetta in cui compare t e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono i termini legati a t da una relazione di tipo NT nel thesaurus, quindi termini più specifici;
- siblings-add(t): propone all'utente una lista di termini da aggiungere nella faccetta in cui compare t e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono i termini "fratelli" di t, ossia gli NT dei BT di t;
- morph-add(t): propone all'utente una lista di termini da aggiungere nella faccetta in cui compare t e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono termini morfologicamente simili a t;
- truncate-add(t): propone all'utente il troncamento del termine t, da aggiungere nella faccetta di t e aggiunge in tale faccetta i termini confermati dall'utente.

Tattiche che sostituiscono termini:

- parallel-ut-subst(t): propone all'utente una lista di termini da cui sceglierne alcuni per sostituire t. Quindi disattiva il termine t e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono i termini legati a t da una relazione di tipo UT nel thesaurus;
- parallel-rt-subst(t): propone all'utente una lista di termini da cui sceglierne alcuni per sostituire t. Quindi disattiva il termine t e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista

compaiono i termini legati a `t` da una relazione di tipo RT nel thesaurus;[3]
- `morph-subst(t)`: propone all'utente una lista di termini da cui sceglierne alcuni per sostituire `t`. Quindi disattiva il termine `t` e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono termini morfologicamente simili a `t`;
- `sub-subst(t)`: propone all'utente una lista di termini da cui sceglierne alcuni per sostituire `t`. Quindi disattiva il termine `t` e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono i termini legati a `t` da una relazione di tipo NT nel thesaurus, quindi termini più specifici;
- `super-subst(t)`: propone all'utente una lista di termini da cui sceglierne alcuni per sostituire `t`. Quindi disattiva il termine `t` e aggiunge in tale faccetta i termini confermati dall'utente. In tale lista compaiono i termini legati a `t` da una relazione di tipo BT nel thesaurus, quindi termini più generali.

Tattiche che disattivano termini:

- `deact(t)`: propone all'utente di disattivare il termine `t`; se l'utente conferma tale termine viene disattivato.

Si osservi come la conoscenza morfologica sia rappresentata nelle tattiche `morph-add`, `morph-subst` e `truncate-add`, le quali utilizzano gli algoritmi di troncamento e di ricerca sul thesaurus di termini con la stessa radice.

Un piano è, come detto, una successione di tattiche. I piani definiti in `IRES` sono i seguenti:

a) tutte le sequenze di tattiche costituite da un'unica tattica;
b) `hr-exp-safe`, costituito dall'applicazione in sequenza delle tattiche `truncate-add`, `parallel-rt` e `morph-add`. Se applicato a termini in faccette positive, è utilizzato nell'espansione della query quando l'obiettivo è 'alto richiamo'; non viene applicato su termini di faccette negative, in quanto l'operazione di troncamento di termini in faccette negative è giudicata pericolosa;
c) `hr-exp-unsafe`, costituito dall'applicazione in sequenza delle tattiche `super-add`, `sub-add` e `siblings-add`. È utilizzato tipicamente nell'espansione della query quando l'obiettivo è 'alto richiamo' (applicato a termini in faccette positive); può però essere utilizzato anche se la direzione di riformulazione è 'restringere', se applicato a termini di faccette negative;
d) `hp-exp-safe`, costituito dall'applicazione in sequenza delle tattiche `parallel-rt` e `morph-add`. È utilizzato nell'espansione della query

[3] Questa tattica, così come le successive `sub-subst` e `super-subst` sono effettivamente implementate in IRES, mentre non sono implementati i criteri che permettano di rendere tali tattiche preferibili rispetto ad altre, in quanto tali criteri sono di difficile determinazione e non ancora ben compresi. Come conseguenza di ciò, gli unici termini che il sistema propone di sostituire sono ricavati dall'applicazione delle tattiche `parallel-ut-subst` e `morph-subst`.

quando l'obiettivo è 'alta precisione': in tal caso si evitano le operazioni di troncamento. Inoltre, è utilizzato su termini appartenenti a faccette negative se la direzione di riformulazione è 'restringere';

e) `hp-exp-unsafe`, costituito dall'applicazione in sequenza delle tattiche `super-add`, `sub-add` e `siblings-add`. È utilizzato nell'espansione della query quando l'obiettivo è 'alta precisione' (su faccette positive) o durante il restringimento (su faccette negative).

La versione attuale di `IRES` non utilizza comunque tutti i piani del punto a): solo quelli costituiti da una delle 5 tattiche `parallel-ut`, `parallel-ut-subst`, `morph-add`, `morph-subst` e `deact` vengono applicati durante il processo di riformulazione. I primi 4 sono utilizzati per cercare di ricondursi a termini controllati nel thesaurus (`parallel-ut` e `morph-add` se la direzione di riformulazione è 'espandere', `parallel-ut-subst` e `morph-add-subst` se la direzione di riformulazione è 'restringere'); il quinto piano (costituito dalla tattica `deact`) è utilizzato per disattivare termini (in faccette positive) se la direzione di riformulazione è 'restringere'.

I nomi dei piani sono scelti in modo da essere mnemonici: `hr` indica alto richiamo, `hp` alta precisione, `exp` indica che la direzione di riformulazione è di espansione e `safe` sta per piani che risultano più affidabili dei corrispondenti `unsafe`, ossia piani che è preferibile applicare prima. Infatti, si è ipotizzato che l'esecuzione delle tattiche `truncate-add`, `parallel-rt` e `morph-add` vada eseguita prima rispetto alle `super-add`, `sub-add` e `siblings-add`.[4]

5.3. Criteri per la scelta dei candidati

In questa sezione sono illustrati in modo sistematico i criteri di preferenza fra candidati, introdotti nella sezione 4.3. Alcuni di tali criteri fanno riferimento al tipo di piano (e quindi indicano quale candidato preferire rispetto agli altri confrontando unicamente i piani di tali candidati), altri fanno riferimento al focus (e quindi utilizzano le sue caratteristiche per determinare l'ordinamento di preferenza) e altri criteri ancora si basano sia sul focus che sul piano.

Tali criteri sono divisibili in due gruppi, il primo relativo ai piani che aggiungono nuovi termini, il secondo relativo al piano che disattiva termini (`deact`). I criteri del primo gruppo sono:

- preferire i piani che aggiungono termini alla query a quelli che disattivano termini, in quanto si è ipotizzato che l'aggiunta di termini definisca meglio il BI dell'utente, al contrario della disattivazione;
- preferire i piani 'safe' a quelli 'unsafe', quindi `hr-exp-safe` ad `hr-exp-unsafe` e `hp-exp-safe` ad `hp-exp-unsafe`, in quanto, come già accennato alla fine del paragrafo precedente, si è supposto che i piani 'safe' siano più affidabili ed efficaci;

---

4 Questa conoscenza sarà rappresentata fra i criteri di scelta dei candidati, si veda la prossima sezione.

- a parità di piano, in espansione, trattare prima le faccette con posting count basso e quindi preferire i candidati i cui termini appartengono a faccette con posting count più basso delle altre. La motivazione per tale scelta è che la faccetta con posting count minore è con maggiore probabilità quella che causa il reperimento di pochi documenti;
- a parità di piano e di posting count delle faccette, lavorare prima su termini con grado di interesse per l'utente alto (si veda la sezione 3.1). Poiché il grado di interesse indica quanto un termine è importante per esprimere il BI dell'utente, è ovviamente opportuno cercare dapprima termini alternativi ai termini più importanti;

I criteri per il piano `deact` sono:

- se si devono disattivare termini (e quindi il piano del candidato è `deact`), preferire i termini con grado di interesse basso, in quanto meno importanti per la definizione del BI;
- se si devono disattivare termini, a parità di grado di interesse preferire i termini troncati (che hanno maggiori probabilità di reperire documenti non rilevanti rispetto ai corrispondenti non troncati);
- se si devono disattivare termini e a parità dei due fattori precedenti, preferire i termini in faccette con posting count basso, in quanto in questo modo si diminuisce senz'altro il limite inferiore del numero di documenti reperiti, e quindi con buona probabilità anche il numero di documenti reperiti effettivamente).

Tali criteri costituiscono una parte rilevante della conoscenza esperta rappresentata mediante regole di produzione nell'EKB di `IRES`, come si vedrà nella sezione 6.

Da quanto detto finora si può forse ricavare l'impressione che l'espansione della query sia trattata in modo più accurato del restringimento. Tale impressione è sostanzialmente corretta ed è motivata da due differenti motivi: il primo è che mentre l'espansione e la definizione sempre più accurata del BI possono procedere di pari passo (aggiungendo nuovi termini in or logico si perseguono entrambi gli obiettivi), ciò non è vero per il restringimento, in quanto l'eliminazione di termini contrasta con la definizione completa del BI e l'aggiunta di termini in faccette negative presenta i noti problemi. Il secondo motivo, di natura pragmatica, è che invece il caso dell'espansione è stato sperimentato più a fondo, in quanto con le collezioni a disposizione di FIRE il problema del reperimento di un basso numero di documenti si presenta raramente[5].

6. Il modulo `IRES`: implementazione

In questa sezione è descritta l'implementazione di `IRES` ad un livello più specifico, facendo riferimento al codice.

[5] È recente l'introduzione in FIRE di una collezione di 20.000 documenti, sulla quale il problema del restringimento potrà in un prossimo futuro essere studiato in modo più approfondito.

L'implementazione di IRES è realizzata utilizzando due differenti linguaggi: LISP e XPSE, un sistema a regole di produzione sviluppato dall'Università di Udine, Dipartimento di Matematica e Informatica, Laboratorio di Intelligenza Artificiale.

Nel paragrafo 6.1 sono presentate le varie fasi seguite da IRES per scegliere di volta in volta le operazioni da effettuare sulla query durante la riformulazione. Nel paragrafo 6.2 è illustrata l'implementazione di tattiche e piani. Nei paragrafi 6.3 e 6.4 sono descritte le strutture dati e le regole utilizzate da IRES. Nei paragrafi 6.5 e 6.6 sono illustrate rispettivamente le interfacce fra IRES e IRESFACE e fra IRES e DBMAN. Infine, il paragrafo 6.7 contiene l'elenco dei file che compongono IRES.

6.1. Flusso di controllo del processo di riformulazione

La riformulazione viene effettuata in varie fasi, illustrate in figura 5 e descritte nel seguito. Ogni fase ha un obiettivo e il suo raggiungimento determina il passaggio alla fase successiva. Il flusso di controllo è gestito con regole di produzione, che eventualmente richiamano funzioni LISP; come conseguenza di ciò, l'obiettivo di ogni fase è quello di asserire nella *working memory* (wm nel seguito) alcuni fatti. Il controllo viene passato da una fase all'altra con il meccanismo dei *contesti*, tipico dei sistemi a regole di produzione (si veda Brownston et al., 1985).

La prima fase, denominata start-ref, ha lo scopo di inizializzare la wm. In pratica, asserisce alcuni fatti che indicano che l'attività di riformulazione è in corso e la direzione di riformulazione (expand se bisogna espandere o narrow se bisogna restringere la query); per fare ciò, effettua una ricerca e

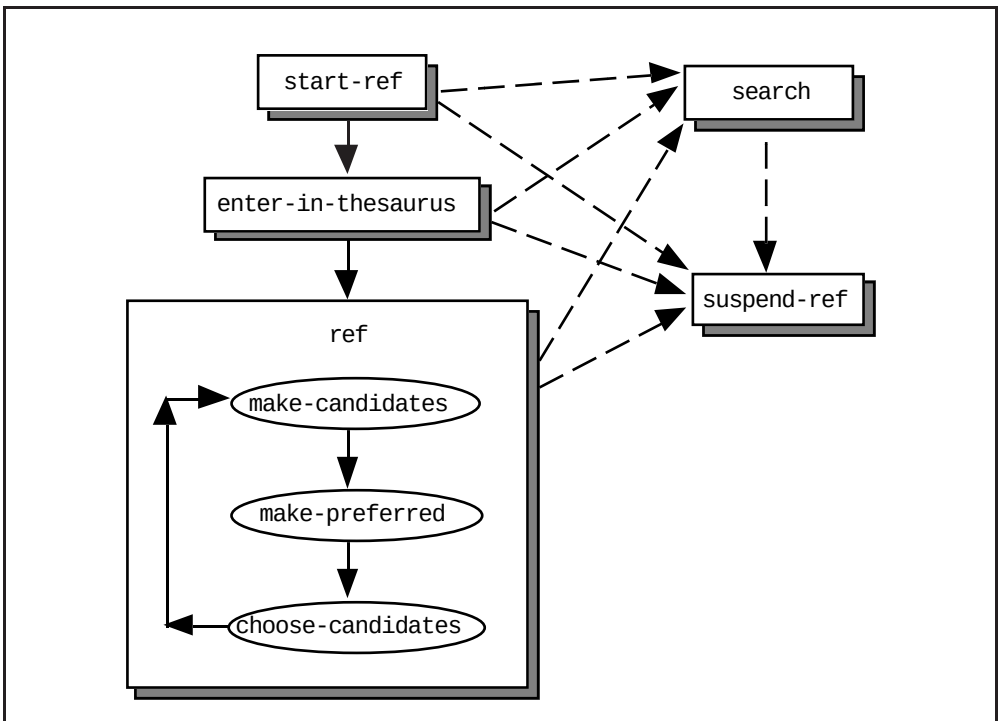

Figura 5. Le fasi del processo di riformulazione. I rettangoli indicano fasi, le ellissi sottofasi, le frecce illustrano il passaggio di controllo da una fase all'altra. Le fasi search e suspend-ref sono in realtà demoni, come descritto nel testo.

confronta il numero di documenti reperiti con l'intervallo di documenti desiderati.

La seconda fase, `enter-in-thesaurus`, è utile nel caso che nella query compaiano termini non appartenenti alla DSKB. Per raggiungere l'obiettivo di questa fase, che è quindi l'introduzione di termini controllati nella query, vengono utilizzati i piani `parallel-ut` e `morph-add` (o `parallel-ut-subst` e `morph-subst` nel caso del restringimento della query): all'utente viene quindi chiesto di confermare l'aggiunta o la sostituzione di termini della query.

La fase successiva, denominata `ref`, esegue un passo di riformulazione, ossia modifica la query eseguendo una sequenza di piani. È suddivisa in 3 sotto-fasi:

- `make-candidates`, che individua tutti i possibili candidati. Per ogni candidato viene asserito un fatto nella wm;
- `make-preferred`, che asserisce nella wm fatti che indicano quale(i) fra i candidati debba essere preferito rispetto agli altri. In questa fase si utilizzano i criteri per la preferenza fra candidati descritti nella sezione 5.3;
- `choose-candidates`, che sceglie uno fra i candidati preferiti (le regole di preferenza dell'EKB definiscono un ordine parziale, non totale, sui candidati) ed applica il piano corrispondente.

Una volta terminata l'esecuzione del piano prescelto, il procedimento viene iterato: il sistema sceglie i nuovi candidati, e così via, finché non si realizzano le condizioni per la terminazione.

In figura 5 compaiono anche altre due componenti: i *demoni* `search` e `suspend-ref`. Essi vengono attivati automaticamente quando nella wm si presentano opportune situazioni, quindi possono accendersi in una qualsiasi delle fasi illustrate in precedenza: questo fatto è rappresentato graficamente dalle frecce tratteggiate. Il demone `search`, che esegue una ricerca nella banca dati con la query attuale e ne memorizza il risultato nella wm, viene attivato quando il numero di modifiche della query effettuate a partire dalla ricerca precedentemente effettuata supera una certa soglia (fissata attualmente a 3 modifiche). `suspend-ref` viene invece attivato quando si verificano le condizioni per sospendere la riformulazione: se la direzione di riformulazione è `expand`, quando il numero di documenti reperiti è maggiore del limite inferiore dell'intervallo di documenti desiderati; se la direzione di riformulazione è `narrow`, quando il numero di documenti reperiti è minore del limite superiore dell'intervallo di documenti desiderati.

6.2. Tattiche e piani

Le tattiche sono implementate con funzioni LISP: ogni tattica è una funzione LISP e la sua esecuzione (ottenuta con la funzione `tactic-apply`) corrisponde con l'applicazione (`apply` del LISP) di tale funzione agli opportuni argomenti. Durante la chiamata di tali funzioni `IRES` presenta all'utente la modifica della query proposta dal sistema ed egli conferma in tutto o in parte tale modifica e questa viene quindi eseguita.

La definizione di piani è realizzata per mezzo della funzione `plan-def`. Esempi di piani definiti in `IRES` sono:

```
(plan-def 'deact '(deact))
(plan-def 'hr-exp-safe '(truncate-add parallel-rt morph-add)),
```

dove la prima riga definisce il piano `deact` composto dall'unica tattica `deact`, e la seconda il piano `hr-exp-safe` composto dalla successione delle tattiche `truncate-add`, `parallel-rt` e `morph-add`.

Essendo un piano una successione di tattiche, la sua esecuzione è ovviamente realizzata tramite l'esecuzione in sequenza delle tattiche che lo compongono. La funzione che si incarica di fare ciò è denominata `plan-apply`.

In questo documento non si scende più in dettaglio nella descrizione dell'implementazione di tattiche e piani. Per ulteriori dettagli si consulti direttamente il codice (file `tactics.lisp`).

6.3. Strutture dati nella WM

In questa sezione sono illustrati i fatti utilizzati da `IRES` per memorizzare le seguenti informazioni nella wm:

- faccette, termini, obiettivo della ricerca e direzione di riformulazione, ricavati direttamente dalla query;
- piani;
- candidati (coppie termine-piano) e preferenze di candidati.

Una faccetta è rappresentata da un insieme di fatti del tipo:

```
(facet ?fc)
(f-sgn ?fc ?sgn)
(f-state ?fc ?state)
(f-pc ?fc ?pc)
(f-lowpc ?fc ?lowpc)
(f-di ?fc ?di)
```

con il seguente significato dei simboli (in XPSE, un oggetto che inizia con un punto di domanda indica una variabile):

- `facet`: `?fc` è un codice di identificazione della faccetta. È usato in `IRES` per identificare in modo univoco una faccetta ed è nascosto all'utente;
- `f-sgn`: segno della faccetta. Può assumere i valori `:positive` (per faccette positive) o `:negative` (per faccette negative);
- `f-state`: flag che indica lo stato della faccetta. Può assumere i valori `:active` (per faccette attive) o `:deactive` (per faccette disattive);
- `f-pc`: posting count (stimato) della faccetta. Nella versione attuale è il massimo posting count dei termini che compaiono nella faccetta, ma si può pensare ad una stima approssimata più precisa basata sui posting count dei termini della faccetta. Con una ricerca nella collezione, il posting count di una faccetta potrebbe essere calcolato in modo esatto; ciò non viene fatto per motivi di efficienza;

- `f-lowpc`: flag che indica se la faccetta ha un posting count sensibilmente più basso delle altre faccette. Assume il valore `:lowpc` per le faccette con posting count basso e `:not-lowpc` per le faccette con posting count alto;
- `f-di`: flag che indica il grado di interesse (*degree of interest*) della faccetta, ricavato effettuando una media dei gradi di interesse dei termini che la compongono. Può assumere i valori `:high` (per indicare un alto grado di interesse) o `:low` (per grado di interesse basso).

Un termine è rappresentato da un insieme di fatti del tipo:

```
(term ?tc ?tn ?fc)
(t-trunc ?tc ?trunc)
(t-owner ?tc ?owner)
(t-di ?tc ?di)
(t-state ?tc ?state)
(t-fields ?tc ?fields)
(t-sl ?tc ?sl)
(t-ct ?tc ?ct)
(t-pc ?tc ?pc)
(t-dskb ?tc ?dskb)
(t-age ?tc ?age)
```

in cui ogni attributo indica rispettivamente:

- `term`: `?tc` è un codice identificativo del termine, `?tn` è il termine vero e proprio (per semplificare le operazioni morfologiche, un termine è rappresentato internamente come lista di stringhe: ad esempio, il termine `"neural networks"` è `("neural" "networks")`) e `?fc` è il codice della faccetta a cui il termine appartiene (nel caso che un termine appartenga a due faccette differenti, vi saranno due fatti di tipo `(term …)` con codici identificativi `?tc` differenti);
- `t-trunc`: flag che indica se il termine è troncato, e in tal caso il valore è una lista del tipo `(truncated <term>)` (in cui `<term>` è il termine originale non troncato) o non troncato (e in tal caso il valore è `:integer`);
- `t-owner`: flag che indica l'origine del termine. Assume il valore `:user` se è un termine introdotto dall'utente, `:derived` se è un termine introdotto durante il processo di riformulazione;
- `t-di`: flag che indica il grado di interesse (*degree of interest*) del termine, introdotto dall'utente. Può assumere i valori `:high` (per indicare un alto grado di interesse) o `:low` (per grado di interesse basso);
- `t-state`: flag che indica lo stato del termine. Può assumere i valori `:active` (per termini attivi) o `:deactive` (per termini disattivi);
- `t-fields`: lista delle sigle dei campi di ricerca del termine, ossia le sigle dei campi della collezione su cui effettuare le ricerche per quel termine;
- `t-sl`: livello di specificità del termine[6];

[6] Il livello di specificità di un termine appartenente al thesaurus è un'indicazione di quanto tale termine sia generale o specifico. È ricavato effettuando una media delle lunghezze dei cammini da tale termine ai nodi radice. Non è per ora utilizzato in IRES, anche se questa informazione potrebbe essere utile durante la concettualizzazione o la riformulazione.

- `t-dskb`: flag di presenza nel thesaurus. Ha valore `:present` se il termine compare nella DSKB, `:not-present` altrimenti;
- `t-ct`: flag che indica se il termine è controllato (valore: `:ct`) o non controllato (valore: `:ut`). Tale flag ha senso solo per termini appartenenti al thesaurus;
- `t-pc`: posting count del termine;
- `t-age`: flag che indica se il termine è stato introdotto nella query durante l'ultimo passo di riformulazione (valore: `:new`) o in precedenza (valore: `:old`).

Altre informazioni memorizzate nella query sono la direzione di riformulazione e l'obiettivo della ricerca, rappresentate con un fatto del tipo:

```
(direction ?dir ?obj)
```

in cui `?dir` può assumere i valori `:exp` (per 'expand') o `:narr` (per 'narrow') e `?obj` può valere `:hp` (per 'high precision') o `:hr` (per 'high recall').

Come detto, tattiche e piani sono implementati tramite funzioni LISP. I piani sono comunque rappresentati anche nella wm, unitamente ad alcune loro caratteristiche, al fine di poter scegliere il piano più appropriato alle circostanze. Per rappresentare i piani si usano fatti del tipo:

```
(plan-name (?dir ?obj ?sgn ?plan-name))
```

dove per ogni piano sono indicati la direzione di riformulazione (`?dir`), l'obiettivo della ricerca (`?obj`) e il segno delle faccette (`?sgn`) per cui è adatto e l'identificatore del piano (`?plan-name`). Ad esempio, alcuni dei fatti che modellano i piani definiti in IRES sono:

```
(plan-name (:exp :hr :positive hr-exp-safe))
(plan-name (:exp :hr :positive hr-exp-unsafe))
(plan-name (:exp :hp :positive hp-exp-safe))
(plan-name (:exp :hp :positive hp-exp-unsafe))
(plan-name (:exp ?#:a :negative deact)).
```

Con `:exp` si indica la direzione di riformulazione 'expand'; il simbolo `?#:a` indica la variabile anonima, ossia una variabile differente ad ogni occorrenza.

I fatti che rappresentano i candidati sono del tipo

```
(candidate ?cc ?tc ?fc ?plan-name)
```

in cui `?cc` sta per il codice identificativo del candidato, `?tc` per il codice del termine a cui applicare il piano di identificatore `?plan-name` e `?fc` per il codice della faccetta in cui tale termine compare. Le preferenze fra i candidati sono modellate con fatti del tipo

```
(not-preferred ?cc),
```

in cui `?cc` è il codice del candidato. Si osservi che non è necessario calcolare tutte le relazioni di ordine fra i candidati: è sufficiente sapere, per ogni candidato, se esso sia o meno un elemento massimale rispetto

all'ordinamento parziale dei candidati (si vedano le sezioni 4.3 e 5.3), informazione ricavabile dai fatti (not-preferred ?cc).

6.4. Criteri per la preferenza dei candidati

I criteri di preferenza dei candidati sono rappresentati mediante regole di produzione, le quali indicano quali piani è preferibile adottare, su quali faccette e su quali termini, basandosi sulla query, sugli obiettivi della ricerca, sul numero di documenti reperiti e sull'intervallo di documenti desiderati (informazioni tutte contenute nella wm).

Nella figura 6 sono presentati alcuni esempi di regole dell'EKB, in sintassi XPSE. Per quanto riguarda la sintassi XPSE, gdefrule definisce una regola, gtell asserisce un fatto nella wm e (unknown ?x) è vero se il fatto ?x non compare nella wm. Infine, il fatto (task …) indica il contesto in cui tale regola viene applicata; in questo caso il contesto è make-preferred (si veda la figura 5). Al fine di ottenere una migliore comprensione, le regole in figura 6 sono una versione leggermente semplificata di quelle che compaiono, insieme ad altre, nel file ref-rule.lisp.

Per facilitare la comprensione delle 3 regole do figura 6, viene presentata, in figura 7, la traduzione in linguaggio naturale della prima regola.

6.5. Interfaccia con IRESFACE

IRESFACE è l'interfaccia utente di FIRE, implementata in CLIM. Le strutture dati utilizzate in IRES appena illustrate non possono essere utilizzate direttamente dal CLIM, che ha invece bisogno di collegare ad ogni elemento visualizzato un oggetto LISP a cui accedere tramite puntatore. A causa di ciò, l'interazione fra IRES e IRESFACE risulta piuttosto macchinosa, in quanto operazioni che potrebbero avvenire in modo automatico, quali il redisplay di alcune parti dell'interfaccia al momento opportuno (quando i valori vengono modificati) devono invece essere gestite da IRES, con un notevole calo della bontà di modularizzazione del sistema.

In un caso ideale, l'aggiornamento delle informazioni presentate all'utente dovrebbe essere compito dell'interfaccia, la quale dovrebbe continuamente monitorare i valori dei dati di interesse per l'utente ed effettuare le operazioni di visualizzazione opportune; in tal modo, IRES sarebbe svincolato da operazioni che non sono probabilmente di sua competenza.

Illustrando più in dettaglio l'interfaccia IRES-IRESFACE, quando l'utente avvia la riformulazione premendo il bottone Reformulate, IRESFACE passa l'intera struttura dati rappresentante la query ad IRES, che si incarica di trasformarla in fatti nella wm, e il processo inverso deve ovviamente avvenire quando IRES comunica ad IRESFACE la nuova query. Le strutture dati utilizzate nella wm per rappresentare la query sono state descritte nella sezione 6.3; il passaggio di dati fra IRESFACE e IRES avviene utilizzando oggetti LISP, più precisamente le strutture illustrate in figura 8 (con i valori di default specificati).

```
(gdefrule PREF-HR-SAFE-PLANS
  ;; Preferire il piano hr-exp-safe al piano hr-exp-unsafe
  (and
   (task make-preferred)
   (direction :exp :hr)
   (candidate ?c1 ?#:a ?#:a hr-exp-safe)
   (candidate ?c2 ?#:a ?#:a hr-exp-unsafe)
   (unknown (not-preferred ?c2)))
   ->
   (gtell (not-preferred ?c2)))

(gdefrule PREF-F-LOWPC
  ;; A parita' di piano, preferire faccette con lowpc
  (and
   (task make-preferred)
   (direction :exp ?#:)
   (candidate ?c1 ?#: ?f1 ?p)
   (candidate ?c2 ?#: ?f2 ?p)
   (f-lowpc ?f1 :lowpc)
   (f-lowpc ?f2 :not-lowpc)
   (unknown (not-preferred ?c2)))
   ->
   (gtell (not-preferred ?c2)))

(gdefrule PREF-T-DI-HIGH
 ;;A parita' di piano e di f-lowpc, preferire termini con di high
  (and
   (task make-preferred 4)
   (direction :exp ?#:)
   (candidate ?c1 ?t1 ?f1 ?p)
   (candidate ?c2 ?t2 ?f2 ?p)
   (f-lowpc ?f1 ?lowpc)
   (f-lowpc ?f2 ?lowpc)
   (t-di ?t1 :high)
   (t-di ?t2 :low)
   (unknown (not-preferred ?c2)))
   ->
   (gtell (not-preferred ?c2))).
```

Figura 6. Alcune regole per la preferenza dei candidati.

Quindi la struttura `cufo-s` contiene una lista di strutture `facet-s` (nel campo `fl` della struttura `cufo-s`) e ognuna delle strutture `facet-s` contiene una lista (in `tl`) di strutture `terms`. A questo punto, il significato dovrebbe essere chiaro se si confrontano queste strutture dati con i fatti della wm descritti nel paragrafo 6.3. La definizione di tali strutture è effettuata nel file `cufo.lisp`.

Le funzioni incaricate di gestire le traduzioni fra fatti nella wm e strutture LISP, entrambe implementate nel file `cufo.lisp`, sono la `get-query`, che legge nella wm i fatti relativi alla query e restituisce la struttura corrispondente, e la `define-query`, che partendo da una struttura asserisce nella wm i fatti corrispondenti.

Inoltre, siccome le modifiche che `IRES` (previa conferma dell'utente) apporta alla query non vengono automaticamente riportate sul `Query Panel` (si veda la figura 2), è necessario che sia `IRES` ad invocare, dopo ogni modifica, un *redisplay* dei *pane* opportuni. Questo viene fatto tramite la

SE:
- il contesto è 'make-preferred' (ossia se si stanno scegliendo i candidati massimali),
- la direzione di riformulazione e l'obiettivo sono rispettivamente 'expand' e 'alto richiamo',
- vi è un candidato ?c1 il cui piano è hr-exp-safe,
- vi è un candidato ?c2 il cui piano è hr-exp-unsafe
- non è noto il fatto che il candidato ?c2 sia non massimale,

ALLORA
asserisci il fatto che il candidato ?c2 non è un candidato massimale.

Figura 7. Traduzione in italiano della prima regola di figura 6.

funzione redisplay, implementata in IRESFACE e chiamata da IRES al bisogno.

6.6. Interfaccia con il DBMAN

Il DBMAN fornisce funzioni a moduli implementati in LISP ed è implementato in C. A causa di ciò si è reso necessario interfacciare opportunamente il DBMAN con il resto del sistema. Tale interfaccia è

```
(defstruct (cufo-s)
  (fl nil)                    ; facet list
  (tactic-done-facts nil)     ; wm control facts list
  (search-obj 'high-recall)   ; high-recall / high-precision
  (docs-int '(10 30))         ; desired docs. range

(defstruct (facet-s)
  (code nil)                  ; Facet internal code
  (sgn :positive)             ; Facet sign
  (state :active)             ; Facet state
                              ; (:active/:deactive)
  (tl nil)                    ; List of terms in this facet
  (pc nil)
  (lowpc nil)                 ; :lowpc / :not-lowpc
  (di nil))                   ; :high / :low

(defstruct (terms)
  (name nil)                  ; List of strings
  (fc nil)                    ; Facet internal code
  (trunc :integer)            ; (truncated tn)/:integer
  (owner :derived)            ; :user/:derived
  (di :high)                  ; :high/:low
  (state :active)             ; :active/:deactive
  (fields '())                ; Term fields list
  (sl nil)                    ; Specificity level
  (ct nil)                    ; :ct / :ut
  (pc nil)                    ; Posting count
  (dskb nil)                  ; :present / :not-present
  (age :new))                 ; :new / :old
```

Figura 8. Strutture LISP utilizzate per il passaggio di dati fra IRES ed IRESFACE.

contenuta nel file `dbman-int.lisp`: in esso è dapprima definita la *foreign function* `dbm`, e poi tale funzione è sfruttata per implementare le funzioni LISP necessarie al resto del sistema. Si veda il file `dbman-int.lisp` per ulteriori dettagli.

6.7. File di IRES

Una versione di FIRE si può trovare nel direttorio:

```
/progetti/fire/PUBLIC-VERSION/fire,
```

a cui ci si riferisce nel seguito con `~fire`.

I sorgenti dei vari moduli che compongono FIRE si trovano nel direttorio `~fire/main`, suddivisi nei seguenti sottodirettori:

```
c-code/ dskbman/ ires/ foldermanager/ new-interface/ stem/ xpse/.
```

I files che compongono IRES sono i seguenti:

- `dskbman-int.lisp`: file di interfaccia fra IRES e il DSKBMAN;
- `collections.lisp`: file contenente le funzioni e le variabili per gestire le varie collezioni su cui FIRE lavora;
- `cufo.lisp`: file contenente le funzioni di interfaccia fra XPSE e LISP. Il suo nome deriva da CUrrent FOrmulation, sinonimo di 'query';
- `ref-rule.lisp`: file contenente le regole dell'EKB;
- `tactics.lisp`: file contenente la definizione delle tattiche e dei piani implementati in IRES;
- `term-functions.lisp`: file contenente funzioni di utilità per termini (posting count, livello di specificità, ecc.), definite utilizzando i servizi di `dskbman-int.lisp` e `dbman-int.lisp`;
- `translate-cufo.lisp`: file per tradurre la query da formato FIRE a formato Fultext;
- `utility.lisp`: file con funzioni di utilità generali.

7. Conclusioni

In questo documento è stato illustrato il modulo IRES, componente esperta del sistema FIRE. Dopo una breve introduzione al campo dell'IR (sezione 2), nella sezione 3 è illustrato a grandi linee il sistema FIRE. La sezione 4 descrive IRES ad un livello concettuale, la sezione 5 ad un livello logico (viene presentata la struttura delle basi di conoscenza del sistema) e la sezione 6 illustra alcuni dettagli implementativi. Per ulteriori informazioni su IRES è necessario consultare direttamente il codice, che comunque è ampiamente commentato.

Per quanto riguarda l'adozione dei piani, è importante osservare che tale meccanismo non sembra soddisfacente dal punto di vista concettuale, anche se ha risolto alcuni problemi di efficienza temporale. Inoltre, la distinzione fra piani 'safe' ed 'unsafe' non sembra sensata.

Per quanto riguarda gli sviluppi futuri previsti per il sistema FIRE, si può fare riferimento a Mizzaro (in preparazione).

Bibliografia

Bates, M. J., 1979. "Information Search Tactics", *Journal of the American Society for Information Science*, July 1979, pp. 205-214.

Belkin, N. J., 1980. "Anomalous states of knowledge as a basis for IR", *The Can. Jou. of Inf. Sci.*, 5, pp. 133-143.

Brajnik G., Guida G., Mastrodonato L., Scaroni C., Tasso C., 1990a. *Progetto Generale del Sistema FIRE: Un Ambiente flessibile per lo Sviluppo di Interfacce Esperte a Sistemi di Basi di Dati Bibliografici*, Rapporto Tecnico CNR, n. 5/43, sottoprogetto Sistemi evoluti per Basi di Dati, Progetto Finalizzato CNR Sistemi Informatici e Calcolo Parallelo.

Brajnik G., Guida G., Mastrodonato L., Scaroni C., Tasso C., 1990b. *Interfacce Cooperative e Flessibili per Utenti di Basi di Dati Bibliografici*, Rapporto Tecnico CNR, n. 5/44, sottoprogetto Sistemi evoluti per Basi di Dati, Progetto Finalizzato CNR Sistemi Informatici e Calcolo Parallelo.

Brajnik G., Guida G., Mastrodonato L., Scaroni C., Tasso C., 1991a, *Specifiche di progetto del modulo per il retrieval intelligente IRES nell'ambito del sistema FIRE*, Rapporto Tecnico CNR, n. 5/61, sottoprogetto Sistemi evoluti per Basi di Dati, Progetto Finalizzato CNR Sistemi Informatici e Calcolo Parallelo.

Brajnik G., Guida G., Mastrodonato L., Scaroni C., Tasso C., 1991b. *Progetto dell'interfaccia utente del sistema FIRE*, Rapporto Tecnico CNR, n. 5/62, sottoprogetto Sistemi evoluti per Basi di Dati, Progetto Finalizzato CNR Sistemi Informatici e Calcolo Parallelo.

Brooks, H.M., 1987. "Expert Systems and Intelligent Information Retrieval", in *Information Processing and Management*, 23 (4), pp. 367-382.

Brownston, L. et al., 1985, *Programming Expert Systems in OPS5. An introduction to rule-based programming*, Addison-Wesley, Reading, MA.

Croft, W. B., 1993. "Knowledge-Based and Statistical Approaches to Text Retrieval", *IEEE Expert*, April 1993.

Danesi, D., 1990. *Le variabili del thesauro - Gestione e struttura*, IFNIA, Laboratorio Thesauri, Firenze.

Floreanini, A., 1993. *Il Progetto FIRE - Studio e Realizzazione di un Sistema di Intelligent Information Retrieval*, Rapporto di Ricerca del Dipartimento di Matematica e Informatica dell'Università di Udine, Udine, Italy, rapporto n. UDMI/17/93/RR.

Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T., 1987. "The Vocabulary Problem in Human-System Communications", *Communications of the ACM*, v. 30, n. 11, November 1987.

Gri, F., 1993. *Analisi delle Funzionalità di Interfacce Intelligenti a Sistemi di Banche Dati Bibliografici*, Tesi di laurea, Università di Udine, Udine, Italy.

Guida G. e Tasso C., 1994. *Design and Development of Knowledge-Based Systems – From Life Cycle to Methodology*, John Wiley and Sons, Chichester, UK.

Ingwersen P., 1992. *Information Retrieval Interaction*, Taylor Graham, London.

Meadow, C. T. and Cochrane, P. A., 1981. *Basic of Online Searching*, John and Wiley, New York, NY.

Mizzaro, S., in preparazione. *Sviluppi futuri di FIRE*, Rapporto di Ricerca del Dipartimento di Matematica e Informatica dell'Università di Udine, Udine, Italy.

Porter, M. F., 1980. An algorithm for suffix stripping, *Program* , 14 (3), pp. 130-137.

Rich E., 1986. *Intelligenza Artificiale*, Mc Graw-Hill, Milano.

Salton, G., 1989. *Automatic Text Processing – The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley.