Personalizzazione di Servizi Tramite Ontologie e Reti Bayesiane

Alessandro Soro

30 marzo 2004

Indice

Elenco delle figure

Capitolo 1

Introduzione

La vastissima disponibilità di dati e servizi cui il Web ci ha abituato negli ultimi anni, ha fatto emergere la necessità di personalizzare tali risorse, adattandole alle specifiche esigenze degli utenti.
Un servizio personalizzato modifica la propria interfaccia o i propri contenuti in base alle preferenze dell'utente, al contesto d'uso, e al dispositivo usato; in modo particolare, le preferenze dell'utente, possono essere prese in considerazione dai recommender-systems, per filtrare le informazioni presentate, dando maggiore enfasi a quelle ritenute più interessanti e penalizzando o eliminando completamente quelle meno rilevanti.
I sistemi di personalizzazione così concepiti necessitano di un modello dell'utente detto User Profile, e di informazioni dettagliate sulle risorse da personalizzare, oltreche di strategie e algoritmi che permettano di prevedere le scelte future in base alle scelte compiute in passato dall'utente, o da utenti simili per profilo o per contesto.

Ad esempio se l'utente di un servizio di libreria virtuale perfeziona l'acquisto di un libro, alcune caratteristiche del prodotto (autore, genere...) possono essere considerate utili indizi per prevedere le scelte future, altre potrebbero essere casuali (anno di pubblicazione, prezzo, numero di pagine...), altre ancora quasi certamente sono ininfluenti ai fini della personalizzazione (codice ISBN, curatore della traduzione...).
Un efficace sistema di personalizzazione deve essere in grado di distinguere i motivi ricorrenti nelle scelte dell'utente, e generalizzare queste informazioni per guidare gli utenti nella scelta dei prodotti e segnalare nuovi titoli che potrebbero interessare.

Il presente lavoro propone un sistema per la personalizzazione di servizi e risorse che usa come modello di dati una ontologia probabilistica, cioè una

ontologia integrata da una rete bayesiana.
L'ontologia fornisce un modello altamente strutturato del dominio di pertinenza del servizio o delle risorse che si desidera personalizzare. Questo modello può supportare inferenze logiche, controlli di integrità, interoperabilità tra servizi eterogenei disaccoppiati, ed ha le caratteristiche di versatilità e riusabilità indispensabili per l'uso nell'ambito di domini in continua evoluzione come i Web-Services.
La rete bayesiana permette di catturare l'incertezza di certe relazioni tra concetti dell'ontologia, e si integra in modo naturale con quest'ultima essendo entrambe rappresentabili come grafi orientati di concetti e relazioni. Il sistema proposto è stato implementato come modulo di personalizzazione nell'ambito del progetto e-MATE, una architettura distribuita per la fornitura di servizi online multicanale, componibili, geo-referenziati e personalizzati.

Capitolo 2

Personalizzazione e User Profiling

Uno degli effetti dell'esplosione del Web è stato l'aumento di interesse per la personalizzazione di servizi e risorse. In generale un servizio personalizzato è un servizio che modifica la propria interfaccia utente o i propri contenuti in funzione di ciò che l'utente specifico richiede. Dal punto di vista dei fornitori di servizi i vantaggi derivanti dall'offrire servizi personalizzati risiedono nel valore aggiunto che i servizi stessi vanno a conquistare, a costi relativamente contenuti: i contenuti personalizzati sono percepiti come più preziosi dai clienti, il che si traduce in maggiori introiti e margini di profitto più consistenti[25]. Poiché inoltre la personalizzazione sarà tanto più efficace quanto più il cliente è un frequentatore abituale del servizio, si può prevedere un beneficio sulla fidelizzazione dell'utente[35]. Ad esempio un motore di ricerca che abbia raccolto nel corso del tempo un profilo dei propri utenti abituali, e utilizzi tale profilo per personalizzare la selezione delle risorse, potrà contare su tale patrimonio informativo per far fronte alla concorrenza di nuovi operatori.
Le motivazioni che spingono gli utenti a preferire servizi personalizzati sono più complesse:

Usability: Con il termine usabilità ci si riferisce spesso alla facilità con cui un utente può imparare a controllare, preparare gli input e interpretare gli output di un sistema o componente software. tale definizione tuttavia non tiene conto di altre importanti caratteristiche quali l'utilità del sistema software (se esso fornisca le necessarie funzionalità), la sua affidabilità, i tempi di risposta ecc.
Più correttamente allora l'usabilità è la misura della profitto che uno specifico utente può trarre dall'uso di un sistema software, nel raggiungimento di un certo obiettivo con efficacia, efficienza e soddisfazione, in uno specifico contesto d'uso. Bevan[3] introduce a tale scopo il termine "qualità d'uso"

come una caratteristica misurabile (in termini appunto di efficacia, efficienza e soddisfazione) dell'interazione tra l'utente e il prodotto. In tal senso non esiste un prodotto "usabile" o uno "non usabile", dato che ad esempio un certo sistema potrebbe essere *non usabile* per utenti inesperti e contemporaneamente *usabile* per utenti addestrati.

Targeting: il Web ha esposto con grande chiarezza quanto sia importante raggiungere tutte e sole le informazioni alle quali siamo interessati. A tal fine i meta-dati svolgeranno un ruolo fondamentale, ma altrettanto importanti saranno le strategie di *matchmaking*, ovvero determinare se una data risorsa corrisponde a una certa descrizione. La prossima generazione di motori di ricerca dovrà essere in grado di rispondere a query complesse, formulate in base alle proprietà delle risorse cercate e non solo a parole chiave presenti sulle pagine stesse. Dovranno essere in grado di trovare una risorsa anche quando questa è distribuita in più documenti fisici, nessuno dei quali, singolarmente, soddisfa la query. E dovranno selezionare i contenuti in base ai criteri soggettivi dell'utente, tenendo in considerazione l'età, la nazionalità, la professione, e le ricerche e le scelte passate.

Pervasive Computing: uno stesso servizio può essere accessibile attraverso vari, differenti, devices[1]. Distribuire un servizio attraverso canali alternativi richiede un alto grado di struttura dei dati e di separazione tra contenuti e presentazione. Uno degli obiettivi della personalizzazione dei servizi dovrebbe essere quello di adattare la presentazione e i contenuti in funzione del device utilizzato per accedervi. Un utente che accede a un servizio attraverso un computer palmare ha certamente diverse esigenze rispetto ad uno che usa un personal computer o una TV interattiva. Il primo si trova probabilmente lontano da casa o dall'ufficio, il secondo quasi certamente in ufficio, in terzo in poltrona. Il primo ha a disposizione un piccolo display, tipicamente 320 240 pixel, il secondo un display ampio ed a elevata risoluzione, il terzo uno schermo televisivo, più ampio ma meno nitido di un monitor per computer. È possibile individuare dei comportamenti ricorrenti in queste classi di utenti che devono essere tenuti in conto nella presentazione dei servizi alterando di volta in volta i modelli di interazione e i contenuti per adattarli alla specifica situazione.

[1]Un tale servizio è di solito chiamato *multicanale*

2.1 Profilo Utente

Con il termine *User Profile* si indica un modello che permette di collegare le informazioni disponibili sull'utente alle aspettative sul suo comportamento futuro[19]. I dati inclusi nel profilo utente sono generalmente distinti in due categorie[1]:

effettivi o demografici comprendono i dati anagrafici, il reddito, gli studi compiuti, la professione... Tali informazioni permettono di classificare un utente come appartenente ad un dato gruppo o di misurare le differenze tra il profilo di vari utenti o gruppi di utenti. Lo scopo è di individuare comportamenti ricorrenti e regole valide per un gruppo di utenti supponendo che le preferenze del gruppo possano essere assunte come approssimazione delle preferenze del singolo utente. Secondo [19] uno dei metodi più efficaci per predire le azioni future di un utente consiste nell'osservare le azioni passate: se in passato l'utente ha eseguito una sequenza di azioni simili a quella appena eseguita, l'azione che l'utente ha eseguito subito dopo in passato è quella che ha le maggiori probabilità di essere eseguita adesso. Classificare l'utente come appartenente ad un gruppo permette di effettuare analoghe deduzioni pur non disponendo di una lunga registrazione della sua storia passata;

transazionali sono l'elenco delle preferenze espresse dall'utente, esplicitamente o implicitamente. Questi dati sono complementari a quelli demografici. Essi costituiscono la storia dell'utente e la loro analisi approfondita permette di individuare pattern ricorrenti nello specifico utenti o in intere classi di utenti. Una conseguenza di ciò è che è possibile ipotizzare il profilo demografico dell'utente in base al suo profilo transazionale[29], cioè stimare con ragionevole accuratezza l'appartenenza dell'utente ad un dato gruppo sulla base del suo comportamento. Il profilo transazionale dovrebbe includere, per essere completo, la maggior quantità possibile di informazioni sulle specifiche transazioni, così da permettere di individuare con accuratezza il contesto in cui esse sono avvenute.

Sulla base di tali dati i sistemi di personalizzazione tentano di prevedere quali prodotti, servizi o risorse siano più graditi ad un certo utente.

2.2 Recommender Systems

Spesso gli utenti di un sistema informatico tendono a compiere sempre le stesse azioni o sequenze di azioni, dunque il comportamento passato dell'utente rappresenta con sorprendente accuratezza una stima del suo comportamento futuro, tale assunto è alla base dei più rudimentali, ma nondimeno efficaci, sistemi di personalizzazione. Tuttavia in molti contesti questo non si verifica: ad esempio un cliente di una servizio online per la vendita di libri non comprerà due volte lo stesso volume. È probabile invece che acquisti un libro simile ad altri libri acquistati in passato, dello stesso genere o dello stesso autore, ma per sfruttare questo dato occorre avere a disposizione un criterio per misurare la similitudine fra due libri. Un'altra possibilità consiste nel suggerire all'utente le scelte compiute da utenti con comportamenti simili. Le strategie adottate per effettuare le previsioni si dividono in *content-based*, che raccomandano risorse simili a quelle che l'utente ha scelto in passato, e *collaborative-filtering*, che raccomandano le risorse scelte da utenti diversi aventi profili simili.

2.2.1 Metodi Content-Based

I metodi di predizione content-based osservano le scelte di un utente e le registrano. Tali metodi richiedono una certa quantità di *intelligenza*, sotto forma di euristiche o modelli probabilistici, per individuare fra le risorse che l'utente ha esaminato (oppure comprato o scaricato...) dei motivi ricorrenti di interesse. Sulla base di questi, il sistema tenta di predire l'interesse dell'utente per altre risorse, valutando la similitudine o la complementarità fra queste ultime e ciò che l'utente ha preferito in passato. Un limite dei sistemi content-based e che essi non tengono in alcun conto le esperienze fatte con altri utenti nel effettuare predizioni, ma limitano le proprie inferenze ai dati raccolti sulle transazioni di uno specifico individuo.

2.2.2 Metodi Collaborative-Filtering

I metodi collaborative-filtering non tengono conto della natura delle risorse ma fanno uso del profilo utente per raccomandare le risorse scelte in passato da utenti con profili simili. La costruzione e la comparazione dei profili utente rappresenta il nodo cruciale di questi metodi che funzionano secondo il principio del *passaparola*. Se esiste un sistema esplicito di rating delle risorse da parte degli utenti, è possibile raggruppare gli utenti che hanno dato voti simili alle stesse risorse. Tuttavia tale feedback rappresenta un impegno che molti utenti non sono disposti a sostenere, anche perché non ne

traggono diretto beneficio. Ancora più difficile è ottenere dagli utenti degli esempi negativi di risorse. Questo costituisce infatti un impegno ancora più gravoso dato che ciò che piace è in genere molto meno di ciò che non piace o non interessa.

2.3 Personalizzazione e Privacy

I sistemi di personalizzazione devono tenere in debito conto la privacy degli individui. I fornitori di servizi personalizzati registrano informazioni sulle preferenze e le abitudini dei loro utenti che devono essere custodite con la massima cura e non devono essere usate al di fuori degli obiettivi per i quali sono state concesse.
In determinate circostanze la presentazione di contenuti personalizzati può essere percepita come una violazione del diritto alla privacy: ad esempio molti utenti potrebbero non gradire di ricevere nella propria mailbox informazioni politiche, mediche o commerciali, solo per il fatto di aver letto un documento o visitato un sito Web su tali argomenti. Meno ancora risulterebbe gradita l'improvvisa e inaspettata apparizioni di tali notizie su un proprio desktop personalizzato. Altri argomenti come la sessualità o la religione richiedono cure ancora maggiori. La legislazione in materia di trattamento dei dati personali è sufficientemente chiara ma per evitare che l'utente percepisca il sistema di personalizzazione come un *impiccione* piuttosto che come un agente utile, sarebbe bene far si che esso funzioni in modo sempre prevedibile e che gli effetti di ogni azione possano essere annullati se necessario.

2.4 Stato dell'Arte

Il maggiore impulso alla ricerca nel campo della personalizzazione arriva dai fornitori e dagli utenti di Web services. Il Web accoppia una quantità di informazioni virtualmente illimitata ad un bacino di utenza senza precedenti, ciò rende possibile per i fornitori di servizi la raccolta di grandi quantità di dati riguardanti le azioni compiute dagli utenti e rende necessario l'uso da parte degli utenti di software personalizzato per gestire l'enorme quantità di informazione disponibile.

Fra gli esempi più noti di servizi web personalizzati si possono citare Ci-

teSeer[2], Amazon.com[3] e My Yahoo![4].

CiteSeer[12][5] è un search engine per articoli scientifici, capace di estrapolare dati rilevanti dai documenti, come autore(i), abstract e citazioni. Il sistema di personalizzazione di CiteSeer permette agli utenti di segnalare al sistema i propri interessi per essere avvertiti della comparsa di nuovi documenti. L'identificazione di articoli interessanti avviene sia in base a dei vincoli sul contenuto (keywords, citazioni), sia in base a misure di similitudine con altri documenti segnalati dall'utente come esempi del proprio campo d'interesse. L'approccio è chiaramente di tipo content-based e la misura della similarità fra documenti è fatta per mezzo della tecnica TFIDF[5] per quanto riguarda il testo, e la tecnica CCIDF[6] per quanto riguarda le citazioni, rispettivamente disusse in [34] e [4].

Amazon.com raccomanda ai propri clienti titoli simili a quelli da loro acquistati in precedenza, adottando dunque una strategia content-based; inoltre, per ogni libro, ne vengono suggeriti altri che spesso sono stati acquistati insieme ad esso da altri clienti. Tale approccio è chiaramente di tipo collaborative-filtering, in cui l'acquisto di un libro è implicitamente assunto come un voto a favore da parte dell'utente.

My Yahoo! adotta un sistema di personalizzazione che permette agli utenti di configurare il servizio secondo le proprie esigenze, determinando i contenuti che saranno proposti. Le possibilità di personalizzazione spaziano dalla scelta del layout della pagina, alla disponibilità di servizi *location aware*, capaci cioè di adattare il proprio funzionamento all'area geografica di provenienza dell'utente (ad es. un servizio di previsioni meteorologiche che mostri i dati relativi alla regione di residenza), alla possibilità di selezionare l'argomento e la frequenza di aggiornamento per news e bollettini[27].
CiteSeer e Amazon.com sono due sono esempi di *recommender system* mentre My Yahoo! è maggiormente orientato alla personalizzazione dell'interfaccia utente.

I temi principali di ricerca nel campo della personalizzazione sono:

Costruzione automatica e non-intrusiva del profilo utente: non sem-

[2] `http://citeseer.nj.nec.com`
[3] `http://www.amazon.com`
[4] `http://my.yahoo.com`
[5] *Term frequency inverse document frequency*
[6] *Common citation inverse document frequency*

pre è possibile ottenere dall'utente i dati personali necessari alla costruzione del profilo demografico, la ragione di ciò è da ricercarsi sia nella sfiducia innata verso servizi troppo curiosi, sia nel costo, in termini di tempo, richiesto dalla compilazione dei relativi moduli. Riuscire a classificare un utente in base al suo comportamento permette di aggirare tale problema. La costruzione del profilo transazionale pone altre, diverse, difficoltà: se il profilo è mantenuto dal servizio esso sarà necessariamente limitato alle interazioni che l'utente ha con il servizio stesso, se invece è mantenuto dall'utente (ad esempio dal Web Browser), si pongono seri problemi di salvaguardia della privacy. Sulla costruzione automatica del profilo cfr. [8] e [29];

Matchmaking di risorse e servizi: offrire servizi personalizzati significa trovare il servizio più adatto dati una serie di parametri che possono essere specificati dall'utenti o inferiti dal contesto. Il problema di individuare una corrispondenza tra la descrizione del servizio e la descrizione dei requisiti è noto come *matchmaking* ed è un tema centrale della ricerca relativa al Semantic Web e al Pervasive Computing, cfr. [9];

Human computer interaction: la personalizzazione non è limitata alla selezione o all'adattamento delle risorse da proporre all'utente, ma comporta anche importanti sforzi nella presentazione delle stesse e nell'interazione con i sistemi informatici, per una discussione sul problema dell'interazione con servizi personalizzati si veda [33];

Algoritmi e strutture dati: Tanto la rappresentazione delle informazioni relative agli utenti e ai servizi, quanto gli algoritmi per la selezione e il ranking dei servizi sono oggetto di ricerca; Si va dall'uso di ontologie (vedi cap. 3) ai modelli probabilistici (vedi cap. 4); una esauriente indagine sullo stato dell'arte per i recommender systems content-based e collaborative si trova in [32];

Capitolo 3

Ontologie

Il significato storico della parola ontologia risale ad Aristotele ed è "Scienza dell'essere" cioè scienza di ciò che esiste. L'ontologia come disciplina filosofica tenta di rispondere a domande come: "Che cosa è l'esistenza?" o "Quali caratteristiche sono comuni a tutti gli esseri?"[26]. Nel contesto informatico tuttavia la parola ontology assume un significato diverso, anche se correlato. Una prima definizione si trova in [15]:

> An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For Knowledge-based systems, what "exists" is exactly that which can be represented.

Dunque la rappresentazione formale di un modello (entità e relazioni) rappresentativo di un certo dominio di interesse. Successivamente tuttavia, la definizione di ontology si è evoluta per comprendere l'aspetto della condivisione di informazioni: in [38] infatti troviamo:

> "Ontology" is the term used to refer to the shared understanding of some domain of interest which may be used as a unifying framework ...

Emerge dunque il concetto di "comprensione condivisa" e quindi un uso delle ontology come supporto all'interoperabilità.
In [37] ci si riferisce alle ontologie in termini più pragmatici:

> An ontology is a document or file that formally defines the relations among terms. The most typical kind of ontology for the Web has a taxonomy and a set of inference rules

ad indicare il ruolo che esse hanno nella ricerca sul Semantic Web[2] come linguaggio per la descrizione e lo scambio di metadati.

La comunicazione è un processo delicato ma necessario tra agenti che desiderino cooperare, siano essi persone, sistemi software o intere organizzazioni. D'altra parte la mancanza di un comune background, la differenza di esperienze e punti di vista o l'uso di simboli diversi per denotare gli stessi concetti, portano spesso a incomprensioni. Per risolvere questo problema è necessario ridurre (o meglio eliminare) l'ambiguità esistente sui termini e concetti usati e giungere ad un comune accordo sul loro significato. Come risultato avremmo dei benefici nei seguenti campi:

Comunicazione tra persone. Persone provenienti da diversi contesti usano infatti spesso terminologie diverse.

Interoperabilità tra strumenti software.

System engineering particolarmente

specification l'identificazione dei requisiti di un sistema software è avvantaggiata dall'uso di una terminologia comune sulla quale si sia raggiunto un accordo.

reliability la terminologia comune rende possibile l'automazione dei controlli di consistenza di un sistema software, garantendo maggiore affidabilità

reusability un sistema le cui specifiche siano state espresse in un linguaggio non ambiguo e compreso da tutte le parti coinvolte potrà più facilmente essere rivalutato alla luce di nuovi requisiti e, se del caso, riutilizzato senza rischio di incompatibilità o inadeguatezza.

Secondo l'ambito di applicazioni dell'ontology avremo allora diversi gradi di formalità nella sua specificazione:

molto informale: espressa in linguaggio naturale, ad esempio un glossario.

semi-informale: espressa in linguaggio naturale strutturato: ad esempio un dizionario.

semi-formale: espressa in un linguaggio artificiale, formalmente definito.

rigorosamente formale: definita rigorosamente in termini di un linguaggio artificiale, con una semantica espressa formalmente e prove di teoremi di completezza e consistenza.

3.1 Struttura di una ontology.

In [26] un'ontology è definita come una 5-tupla

$$O \quad (C; R; H^C; rel; A^O)$$

in cui:

C e R sono due insiemi disgiunti i cui elementi sono rispettivamente gli identificatori dei *concetti* e delle *relazioni*;

H^C è una relazione transitiva $H^C \quad C \quad C$ anche chiamata gerarchia dei concetti o *tassonomia*. $H^C(C_1; C_2)$ indica che C_1 è un sotto-concetto di C_2

una funzione $rel : R ! C \quad C$ che esprima le relazioni di tipo non tassonomico tra concetti.

- La funzione $dom : R ! C$ è il dominio di una relazione $R \; 2 \; R$: $dom(R) \quad Q_1(rel(R))$;
- La funzione $range : R ! C$ è il range di una relazione $R \; 2 \; R$: $range(R) \quad Q_2(rel(R))$;

la scrittura $rel(R) = (C_1; C_2)$ è equivalente a $R(C_1; C_2)$.

un insieme di *assiomi* A^O espressi in un appropriato linguaggio, ad esempio la logica predicativa;

A ciò occorre aggiungere un *lessico* che collega l'ontology ad opportuni insiemi di nomi che identificano i concetti e le relazioni, formalmente un lessico e una 4-tupla:

$$L \quad (L^C; L^R; F; G)$$

in cui

L^C è l'insieme delle *lexical entries* per i concetti;

L^R è l'insieme delle *lexical entries* per le relazioni;

F è una relazione $F \quad (L^C \quad C)$ che accoppia elementi del lessico ai relativi concetti;

- $F(L) = \{C \; 2 \; C | (L; C) \; 2 \; F\}$
- $F^{-1}(C) = \{L \; 2 \; L^C | (L; C) \; 2 \; F\}$

G è una relazione G (L R R) che accoppia elementi del lessico alle relative relazioni;

- G(L) = f R 2 Rj (L; R) 2 Gg
- G 1(R) = f L 2 L Rj(L; R) 2 Gg

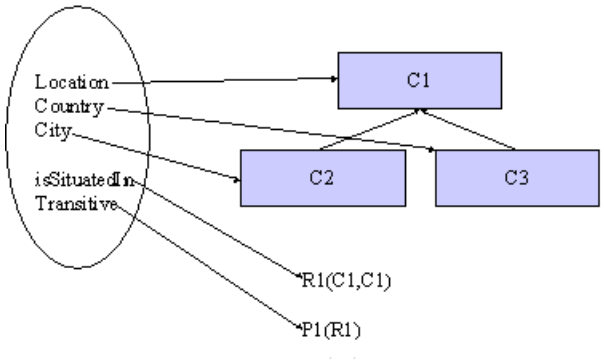

Figura 3.1: Esempio di ontology structure e lessico.

In questo esempio:

l'ontology structure consiste di

C = f C1; C2; C3; P 1g
R = f R1g
H C = f (C2; C1); (C3; C1); (R1; P 1)g
rel = f (R1; (C1; C1))g
A O = 8x; y; z 2 C; R 2 R (xRy ^ yRz ^ P 1(R)) xRz)

Il lessico consiste di:

L C = f Location; Country; City; Transitive g
L R = f isSituatedIn g
F (Location) = C1;
F (City) = C2;
F (Country) = C3;
F (Transitive) = P 1

G(isSituatedIn) = R1

3.2 Metodologie di Design

Al momento non esiste una metodologia generalmente accettata per il design di ontologie. Tuttavia dall'esame di vari lavori sull'argomento (cfr. ad esempio [10], [38], [20], [36]) è possibile individuare un insieme di operazioni da compiere e linee guida per la gestione del processo di sviluppo, raffinamento e valutazione.
Il design di una ontologia comporta almeno quattro distinte fasi, ognuna delle quali comporta l'esecuzione di diverse operazioni.

3.2.1 Specifica dei requisiti

Questa fase avrà tipicamente come risultato la redazione di un documento che specifica:

- il dominio di cui l'ontologia deve fornire un modello e una indicazione della granularità che si desidera raggiungere nella descrizione;
- l'ambito di applicazione dell'ontologia, tramite casi d'uso comprendenti le applicazioni e gli utenti che useranno l'ontologia che si sta sviluppando;
- Il grado di formalità dell'ontologia, dal quale dipende, fra l'altro, la scelta del formalismo di rappresentazione e degli strumenti di sviluppo;
- linee guida per il design, incluse le convenzioni per i nomi dei concetti ecc...;
- fonti di informazione: libri, manuali, esperti, database, interviste, brainstorming ecc...;

Tale fase richiede la collaborazione di esperti del dominio dell'ontologia e di esperti nella modellazione.
Uno strumento particolarmente efficace in questa fase è costituito dalle *competency questions*.
Le competency questions sono una specifica informale dei requisiti dell'ontologia espressa sotto forma di domande alle quali l'ontologia deve permettere di dare una risposta. In generale ogni sostantivo compreso nelle competency

questions deve corrispondere a un concetto nell'ontologia e ogni verbo deve avere come controparte una relazione nell'ontologia[1].

3.2.2 Acquisizione della Conoscenza

In tale fase si esplorano le fonti individuate nel passo precedente per costituire un insieme iniziale di concetti che successivamente formeranno la vera e propria ontologia. Il risultato di tale fase è tipicamente un modello molto informale del dominio in cui i vari concetti individuati sono legati da nessi che specificano l'esistenza di analogie tra di essi. In [38] si suggerisce l'uso di una procedura *middle-out* durante questa fase.
Un approccio *bottomup* ha come risultato una elevata granularità del modello, che rende meno evidente l'esistenza di pattern comuni e caratteristiche ricorrenti nei concetti, rendendo lo sviluppo e il raffinamento dell'ontologia generalmente più laborioso.

Una procedura *topdown*, che a partire da concetti generali tenti di individuare concetti via via più specifici, tende a dare una struttura innaturale al modello: l'individuazione di concetti generali e la loro ripartizione in concetti più specifici deve avvenire mediante raffinamenti successivi durante lo sviluppo dell'ontologia e non essere arbitrariamente imposta.

Un approccio middleout individua dapprima i concetti più importanti del dominio da modellare e procede iterativamente, individuando sottoconcetti che ne specificano meglio le caratteristiche, e precisando somiglianze e differenze quando queste risultano evidenti.

La conoscenza così acquisita risulta organizzata in un modello semistrutturato è può essere sottoposta alla fase successiva di raffinamento.

3.2.3 Concettualizzazione e Codifica

La fase di concettualizzazione raffina il modello semi-formale prodotto nella fase di acquisizione della conoscenza in una vera e propria ontologia. Ciò significa almeno riorganizzare i concetti in una o più tassonomie e aggiungere ad ogni concetto e ad ogni relazione una opportuna descrizione (testuale) di ciò che esso rappresenta del dominio. A seconda del grado di formalità che si è deciso per l'ontologia possiamo inoltre:

[1] Ad es. nello sviluppo di una ontologia del personale di una azienda abbiamo formulato la seguente competency question:
`Quanti` *`impiegati lavorano`* `in un certo` *`dipartimento`*`?`
Ciò significa che gli utenti dell'ontologia useranno il sistema per trovare la risposta a domande di questo tipo e l'ontologia deve definire concetti e relazioni adatti, in questo caso: Impiegato, Dipartimanto , $\text{lavora}_{(\text{Impiegato;Dipartimento})}$.

formalizzare il *domain* e *range* di tutte le relazioni;

definire altre proprietà delle relazioni come massima e minima cardinalità, transitività, ecc...;

definire i rapporti reciproci tra classi, ad es. disgiunzione o identità;

definire l'esistenza di relazioni tassonomiche tra relazioni;

assiomi che definiscano dei vincoli all'uso consistente dei concetti e delle relazioni;

Una volta definito il modello concettuale è possibile procedere con la fase di codifica dell'ontologia nel linguaggio di rappresentazione scelto.Tale operazione, disponendo di un adeguato strumento di sviluppo, sarà effettuata in modo totalmente automatico.

3.2.4 Valutazione

La fase di valutazione ha come obiettivo stabilire se l'ontologia appena realizzata soddisfa tutti i criteri e i vincoli specificati nel corso della prima fase. In particolare risulta molto utile riformulare le competency questions sotto forma di query da eseguire sull'ontologia. Tale operazione potrà far emergere diversi problemi.

L'ontologia potrebbe essere inadeguata per la formulazione della query (ad esempio a causa della mancanza di alcuni concetti o relazioni); in tal caso è necessario reiterare i passi di acquisizione della conoscenza e di concettualizzazione per colmare le lacune e integrare i nuovi concetti.

L'esecuzione delle query può dare risultati scorretti o inaspettati. In tal caso è necessario ripetere la fase di concettualizzazione per eliminare le inconsistenze.
In particolare è possibile creare ontologie perfettamente coerenti ma non soddisfabili da alcun insieme non vuoto di istanze: si consideri il seguente esempio tratto da [11]

Il modello in fig. 3.2 può essere così riassunto:

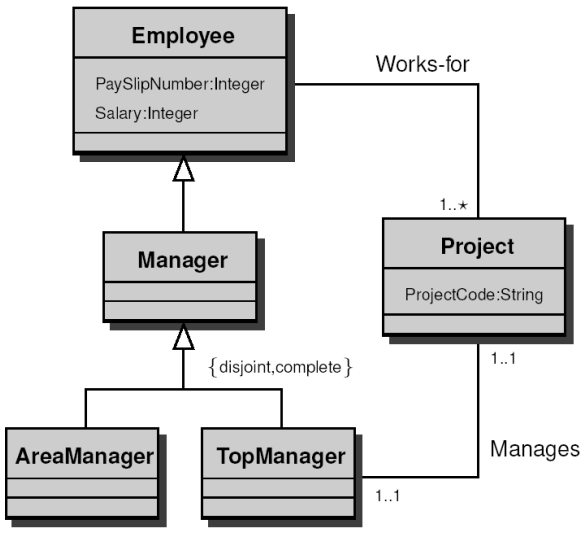

Figura 3.2: Modello per Impiegati e Manager

$$
\begin{array}{lll}
\forall x,y: & \text{Works-for}(x,y) & \Rightarrow \text{Employee}(x) \wedge \text{Project}(y) \\
\forall x,y: & \text{Manages}(x,y) & \Rightarrow \text{Top-manager}(x) \wedge \text{Project}(y) \\
\forall y: & \text{Project}(y) & \Rightarrow \exists x: \text{Works-for}(x,y) \\
\forall y: & \text{Project}(y) & \Rightarrow \exists^{=1} x: \text{Manages}(x,y) \\
\forall x: & \text{Top-manager}(x) & \Rightarrow \exists^{=1} y: \text{Manages}(x,y) \\
\forall x: & \text{Manager}(x) & \Rightarrow \text{Employee}(x) \\
\forall x: & \text{Manager}(x) & \Rightarrow \text{Area-manager}(x) \_ \text{Top-manager}(x) \\
\forall x: & \text{Area-manager}(x) & \Rightarrow \text{Manager}(x) \wedge : \text{Top-manager}(x) \\
\forall x: & \text{Top-manager}(x) & \Rightarrow \text{Manager}(x)
\end{array}
$$

Consideriamo anche come vincolo aggiuntivo:

$$\forall x: \text{Manager}(x) \Rightarrow \forall y:: \text{Works-for}(x,y) \quad (1)$$

ad indicare il fatto che i Manager non lavorano ai progetti ma si limitano a dirigerli.

Si consideri adesso il modello modificato in fig. 3.3, esso aggiunge come ulteriore vincolo il fatto che ogni impiegato lavora ad (almeno) un progetto. Questo nuovo vincolo può essere espresso come:

$$\forall x: \text{Employee}(x) \Rightarrow \exists y: \text{Works-for}(x,y) \quad (2)$$

Ma i vincoli (1) e (2) sono in conflitto dato che ogni Manager è (tramite la

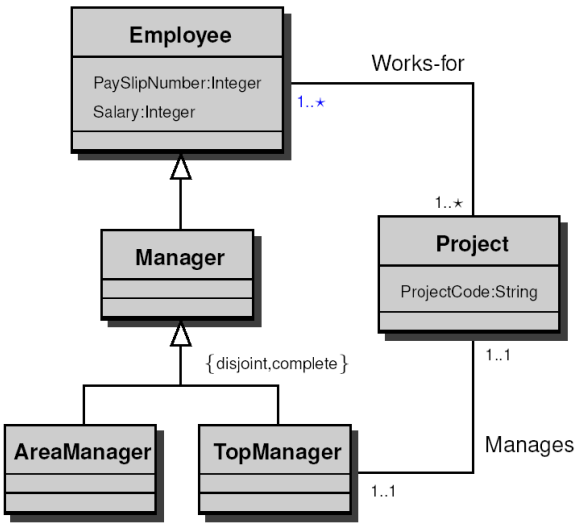

Figura 3.3: Modello non soddisfacibile

relazione tassonomica) anche un Employee. Purtuttavia il modello è coerente ed è soddisfacibile fintanto che tutte le classi considerate sono vuote.

3.3 Design e Valutazione di Tassonomie.

Uno degli elementi che costituiscono una Ontology è la tassonomia o gerarchia di concetti. Essa esprime le relazioni di tipo "is-A" ("è un:") fra concetti e permette di compiere alcune basilari inferenze a partire dai concetti espressi nella Ontology, ossia di derivare nuovi concetti partendo da quelli espliciti e dalle relazioni tra essi. In particolare, quando tra due concetti e vale la relazione tassonomica:

si dice che è un sotto-concetto di e la notazione (x) è usata sia per indicare "x possiede la proprietà " che "x appartiene alla classe ".
La costruzione di una tassonomia coerente non è tuttavia un'impresa priva di ostacoli. Per la gran parte in letteratura si trovano indicazioni di carattere pratico su come riconoscere la relazione tassonomica (applicando appunto in modo intuitivo la relazione is-A) e su come decidere se una data relazione identifichi o meno una classe, cioè a quale dei due possibili significati di (x) stiamo facendo riferimento.

Ad es. sembra ragionevole interpretare la scrittura rosso(Barolo) come "Il Barolo appartiene alla classe dei vini rossi" mentre rosso(Semaforo) non sembra ragionevolmente potersi riferire ad una ipotetica "Classe di tutti i semafori che in questo istante sono rossi". L'inutilità di una tale classe appare evidente dato che:

> Il fatto di essere appunto, "semafori rossi", è l'unica proprietà che accomuna fra loro gli elementi della classe.
>
> Un modo per distinguere un "semaforo rosso" da un altro è in base alla collocazione geografica, o ad un qualche codice di altro tipo; entrambe le cose comunque non anno nulla che fare con il fatto che i semafori siano rossi o meno.

Per contro la classe dei "Vini rossi" raggruppa elementi simili per colore, sapore, abbinamenti gastronomici ecc. É possibile distinguere un "Vino rosso" da un altro in base a elementi come vitigno, bouquet, regione di provenienza, che pur non duplicando la classe "Vino rosso" le sono strettamente legati.

Un insieme di regole formali per valutare la coerenza dell'uso della relazione tassonomica è dato in [17, 18, 16] . Piuttosto che focalizzare l'attenzione sulla semantica della relazione tassonomica, è possibile dare questa per scontata e assumere che l'affermazione " generalizza " o " presuppone " indichi che necessariamente

$$8x;\ (x)\ !\ \quad (x)$$

spostando dunque l'attenzione sulla natura ontologica degli argomenti e . In particolare è utile esaminare i concetti e tenendo conto dei concetti di identità, unità ed essenza. Il concetto di identità permette di distinguere l'uno dall'altro gli elementi di una classe. Il concetto di unità permette di distinguere le parti di una data istanza dal resto del mondo per mezzo di una relazione che le lega. Una volta identificata una data istanza, è di solito possibile riconoscerla a distanza di tempo grazie al fatto che certe proprietà dell'istanza non variano. Altre proprietà possono invece mutare liberamente senza mettere in difficoltà il nostro criterio di identità; per le proprietà che possono cambiare da quelle che non possono (essenziali) è utile il concetto di essenza e il concetto collegato di rigidità di una proprietà.

Rigidità e Proprietà essenziali.

Definiamo una proprietà essenziale per una certa istanza se tale proprietà deve valere necessariamente per quella istanza (indipendentemente dal tempo

e in ogni mondo possibile) Ad es. per un uccello "avere le ali" è una proprietà essenziale dato che essa vale per un uccello in ogni possibile tempo.

> Una proprietà è rigida se essa è essenziale per tutte le istanze per la quale vale.
>
> Una proprietà è non-rigida se e solo se esiste almeno un istanza per la quale essa non è essenziale. Ad es. la proprietà "avere le ali", pur essenziale per gli uccelli, non è una proprietà rigida dato che per le farfalle essa non è essenziale.
>
> Una proprietà è anti-rigida se e solo se essa non è essenziale per tutte le istanze che la posseggono. La proprietà "studente" è anti-rigida dato che per nessuna istanza essa risulta essenziale (è possibile cessare, anche solo temporaneamente di essere studenti e anche se ciò non avviene, la semplice possibilità rende la proprietà non essenziale).

In accordo con la notazione proposta in [16] indicheremo con $+R$ le proprietà rigide, R le proprietà non rigide e $-R$ le proprietà anti-rigide. Si noti che, mentre è possibile che una proprietà rigida generalizzi una proprietà anti-rigida

$$8x; \text{studente}^{R}(x) \; ! \quad \text{persona}^{+R}(x)$$

non è possibile che una proprietà anti-rigida generalizzi una proprietà rigida.

Identità e Identity conditions.

Possiamo affermare che due istanze sono identiche, in rapporto a una data proprietà se e solo se per tale proprietà esiste un criterio di identità r e tale criterio risulta verificato per le istanze date.

$$(x) \wedge \quad (y) \; ! \quad (\; (x; y) \; \$ \; \; x = y)$$

Un modo più efficace di esprimere il criterio di identità è il seguente

$$(x) \wedge \quad (y) \; ! \quad (8z(\; (x; z) \; \$ \quad (y; z)) \; \$ \; \; x = y) \tag{3.1}$$

In cui l'identità delle istanze x e y è ricondotta all'esistenza di una relazione con un terzo elemento z. Ad es. se la proprietà viene interpretata come è-un-insieme e come appartiene-a la 3.1 indica che due insiemi sono identici se e solo se hanno gli stessi elementi.
È anche possibile estendere la 3.1 per tenere conto del tempo

$$9(x;t) \wedge 9(y;t') \wedge (x;t) \wedge (y;t') \rightarrow (8z((x;z;t) \$ (y;z;t')) \$ x = y) \tag{3.2}$$

distinguendo tra identità sincrona nel caso in cui $t = t'$ e diacronica nel caso in cui $t \neq t'$.

Usando sempre la terminologia introdotta in [16]:

> una proprietà trasporta una condizione di identità locale se tale condizione vale solo tra istanze di quella proprietà;
>
> una proprietà trasporta una condizione di identità globale se tale condizione vale anche come criterio di identità verso istanze di altre proprietà cioè se:
>
> $$9(x;t) \wedge 9(y;t') \wedge (x;t) \rightarrow (8z((x;z;t) \$ (y;z;t')) \$ x = y)$$
>
> in tal caso una proprietà è chiamata un sortal ed è marcata dalla meta-proprietà $+I$.
>
> una proprietà fornisce una condizione di identità (locale o globale) se trasporta e non è trasportata anche da tutte le proprietà che generalizzano (quindi se la condizione di identità non viene ereditata, o se viene ereditata solo da alcune proprietà che generalizzano ma non da tutte). Una proprietà che fornisce un condizione di identità locale è marcata +L, una che fornisce una condizione di identità globale è marcata +G;

Secondo [16] una proprietà è un tipo se e solo se essa fornisce un condizione di identità globale.

Questa affermazione ha diverse importanti conseguenze:

> se due entità sono identiche (cioè se sono descrizioni che si riferiscono alla medesima entità) esse devono essere istanze di un sortal comune che trasporta la condizione di identità che esse soddisfano;
>
> sebbene una condizione di identità possa essere trasportata da diversi sortal essa deve essere fornita da un solo sortal che generalizza tutti gli altri;
>
> in una Ontology ben fondata i tipi sono proprietà rigide;
>
> proprietà che trasportano condizioni di identità incompatibili devono essere disgiunte (non possono generalizzare una stessa proprietà)

Unità e Unity conditions.

Il problema dell'unità ha a che fare con l'identificazione delle parti di un'entità: a un certo istante, una data entità, risulta essere un intero (whole), rispetto a una relazione di equivalenza R (che chiamiamo relazione di unificazione) se ogni sua parte risulta legata a tutte le altre, e a niente altro, per mezzo della relazione R.

$$8y(P(y;x;t) ! 8\ \ z(P(z;x;t) \$\ \ R(z;y;t)))$$

Dove $P(y;x;t)$ indica y è parte di x all'istante t.

La relazione R può allora assumere diversi significati, ad esempio:

unità topologica (un blocco di marmo, una zolla di terra);

unità morfologica (una costellazione, una sfera);

unità funzionale (un martello, un bikini);

Una entità x è un intero essenziale (essential whole) rispetto a R se necessariamente essa è sempre un intero rispetto a tale relazione. Una proprietà trasporta un unity condition se esiste un comune relazione di unificazione R tale che tutte le istanze di sono interi essenziali rispetto a R. In tal caso la proprietà sarà indicata con $^{+R}$, altrimenti ^{R}. Una proprietà di cui nessuna istanza è un intero essenziale (cioè non necessariamente) trasporta anti-unità ed è marcata $^{-R}$.

Condizioni di identità, condizioni di unità e rigidità delle proprietà pongono dei vincoli al modo in cui una proprietà (o un tipo) può specializzarne un'altra:

Proprietà con identity conditions e/o unity conditions incompatibili devono essere disgiunte;

Una proprietà anti-rigida non può generalizzare una proprietà rigida.

Una proprietà che trasporta identity conditions non può generalizzare una proprietà che non ne trasporta.

Una proprietà che trasporta unity conditions non può generalizzare una proprietà che non ne trasporta.

Una proprietà che trasporta anti-unity non può generalizzare una proprietà che trasporta unity-conditions.

Tali regole sono di aiuto in numerosi casi pratici di generalizzazioni incoerenti. Ad es. la relazione `oggetto-fisico ! quantita-di-materiale`, che a prima vista può sembrare corretta, nasconde una generalizzazione incoerente. Infatti la proprietà `avere-una-certa-forma` che è essenziale, ad esempio per un vaso, è anti-essenziale per una generica quantità di materiale. La relazione corretta in questo caso è: `costituito-da(oggetto-fisico, quantita-di-materiale)`

3.4 Linguaggi per la descrizione di ontology.

3.4.1 Resource Description Framework (RDF).

RDF è uno strumento proposto dal W3C per l'elaborazione di meta-dati. Esso consiste di un modello e di una sintassi per la serializzazione mediante XML atti a descrivere proprietà e relativi valori, sia come attributi applicabili a risorse, sia come relazioni esistenti fra risorse.

Il modello di dati di RDF è fondato su tre basi:

Resources gli oggetti che si vuole descrivere sono chiamati "risorse", una risorsa può essere qualsiasi cosa cui sia possibile assegnare un URI: da una pagina Web ad un intero sito, da un libro a un individuo. I seguenti sono esempi di URI validi che si riferiscono ad altrettante risorse descrivibili grazie ad RDF:

`http://www.crs4.it`
Il sito Web del CRS4;

`http://www.crs4.it/~asoro/index.html`
La home page di Alessandro Soro nel sito Web del CRS4;

`http://www.crs4.it/~asoro/index.html#author`
L'anchor "author" nella pagina "index.html" nella pagina Web di cui sopra;

Properties Una proprieta è un attributo, una caratteristica o una relazione usata per descrivere una certa risorsa; Le proprietà permettono di associare risorse a particolari valori e ad altre risorse, formando così una rete semantica. Le proprietà delle properties come il tipo della risorsa alla quale una data property è applicabile (domain), i vincoli sul valore associato alla proprietà (range) o le relazioni tassonomiche fra risorse (sub-property)) possono a loro volta essere espresse in RDF mediante RDF Schema;

Statements Uno statement consiste in una risorsa unita mediante una property ad un valore (Literal) o ad un'altra risorsa; I tre elementi di uno statement prendono rispettivamente il nome di subject, predicate e object dello statement;

XML per la serializzazione di RDF.

É possibile serializzare RDF usando XML [6]. La tipica struttura ad albero di un file XML permette di raggruppare le proprietà relative ad una risorsa in modo che la risorsa occupi la radice dell'albero ed ogni foglia rappresenti il valore di una data property per quella risorsa.

La seguente grammatica in EBNF mostra la sintassi di un file RDF.

```
 [1] RDF            ::= ['<rdf:RDF>'] description* ['</rdf:RDF>']
 [2] description    ::= '<rdf:Description' idAboutAttr? '>' propertyElt*
                        '</rdf:Description>'
 [3] idAboutAttr    ::= idAttr | aboutAttr
 [4] aboutAttr      ::= 'about="' URI-reference '"'
 [5] idAttr         ::= 'ID="' IDsymbol '"'
 [6] propertyElt    ::= '<' propName '>' value '</' propName '>'
                      | '<' propName resourceAttr '/>'
 [7] propName       ::= Qname
 [8] value          ::= description | string
 [9] resourceAttr   ::= 'resource="' URI-reference '"'
[10] Qname          ::= [ NSprefix ':' ] name
[11] URI-reference  ::= string, interpreted per [URI]
[12] IDsymbol       ::= (any legal XML name symbol)
[13] name           ::= (any legal XML name symbol)
[14] NSprefix       ::= (any legal XML namespace prefix)
[15] string         ::= (any XML text, with "<", ">", and "&" escaped)
```

Una description contiene uno o più Statements relativi ad una stessa Resource, ad esempio lo statement illustrato in figura 3.4.1 può essere serializzato nel seguente file XML:

```
<?xml version="1.0"?>
  <rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:s="http://description.org/schema/">
    <rdf:Description about="http://www.w3.org/Home/Lassila">
      <s:Creator>Ora Lassila</s:Creator>
    </rdf:Description>
  </rdf:RDF>
```

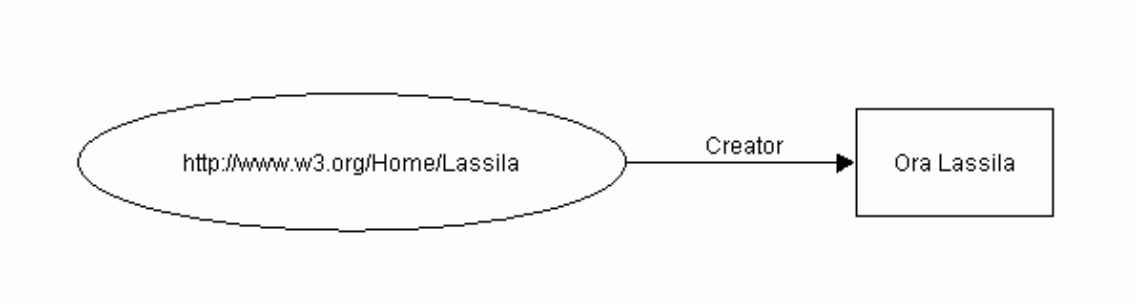

Figura 3.4: Esempio di Statement RDF.

Containers

In RDF è possibile fare riferimento a gruppi di risorse raccogliendo queste ultime in Containers; ne esistono tre tipi:

Bag è una lista non ordinata di risorse o literal, da usare quando l'ordine delle risorse non è significativo. Una stessa risorsa può essere contenuta più volte;

Sequence Come una Bag ma preserva l'ordinamento degli elementi contenuti;

Alternative Una lista di risorse o literal che rappresenta le possibili alternative per il valore di una property;

Per assegnare elementi ad un container è necessario dichiarare una istanza di uno dei tipi di container descritti (per mezzo della property *rdf:type*), e quindi associare le risorse al container come valori di particolari property chiamate $rdf:\_1$, $rdf:\_2$, $rdf:\_3$ ecc.

Ecco in EBNF la grammatica relativa ai container:

```
[18] container       ::= sequence | bag | alternative
[19] sequence        ::= '<rdf:Seq' idAttr? '>' member* '</rdf:Seq>'
[20] bag             ::= '<rdf:Bag' idAttr? '>' member* '</rdf:Bag>'
[21] alternative     ::= '<rdf:Alt' idAttr? '>' member+ '</rdf:Alt>'
[22] member          ::= referencedItem | inlineItem
[23] referencedItem  ::= '<rdf:li' resourceAttr '/>'
[24] inlineItem      ::= '<rdf:li>' value '</rdf:li>'

[1a] RDF             ::= '<rdf:RDF>' obj* '</rdf:RDF>'
```

```
[8a] value          ::= obj | string
[25] obj            ::= description | container
```

Il seguente esempio illustra l'uso dei Container per raggruppare risorse:

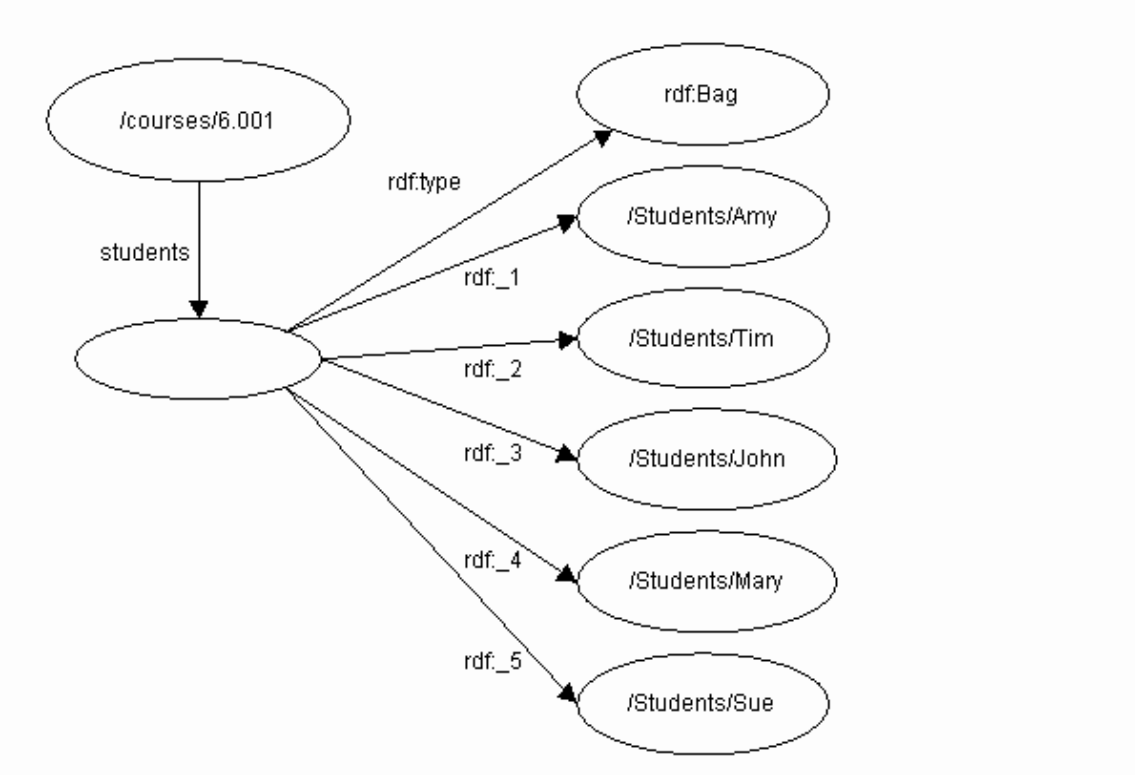

Figura 3.5: Esempio di container RDF.

Il modello illustrato in figura 3.4.1 è rappresentato dal seguente file RDF:

```
<rdf:RDF>
    <rdf:Description about="http://mycollege.edu/courses/6.001">
      <s:students>
        <rdf:Bag>
          <rdf:li resource="http://mycollege.edu/students/Amy"/>
          <rdf:li resource="http://mycollege.edu/students/Tim"/>
          <rdf:li resource="http://mycollege.edu/students/John"/>
          <rdf:li resource="http://mycollege.edu/students/Mary"/>
          <rdf:li resource="http://mycollege.edu/students/Sue"/>
```

```
        </rdf:Bag>
      </s:students>
    </rdf:Description>
  </rdf:RDF>
```

Reification

É possibile costruire un modello di uno statement e fare riferimento a tale modello come ad una risorsa, tale operazione è chiamata *Reification* e il modello di uno statement è chiamato *reified statement*. Tramite reification è possibile esprimere statement riguardo ad altri statement.

Un reified statement consiste di una risorsa con quattro properties:

subject il subject dello statement al quale si desidera fare riferimento;

predicate il predicate dello statement;

object l'object del reified statement;

type il tipo della risorsa che stiamo definendo (il reified statement), deve essere rdf:Statement;

tale descrizione nel suo complesso identifica una risorsa che a sua volta può essere subject o object di properties.

Ad esempio l'affermazione *"Ralph Swick says that Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila"* può essere modellata con un reified statement nel modo indicato in figura 3.4.1

e serializzato in XML come:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:a="http://description.org/schema/">
    <rdf:Description>
      <rdf:subject resource="http://www.w3.org/Home/Lassila" />
      <rdf:predicate resource="http://description.org/schema/Creator" />
      <rdf:object>Ora Lassila</rdf:object>
      <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement"
      <a:attributedTo>Ralph Swick</a:attributedTo>
    </rdf:Description>
  </rdf:RDF>
```

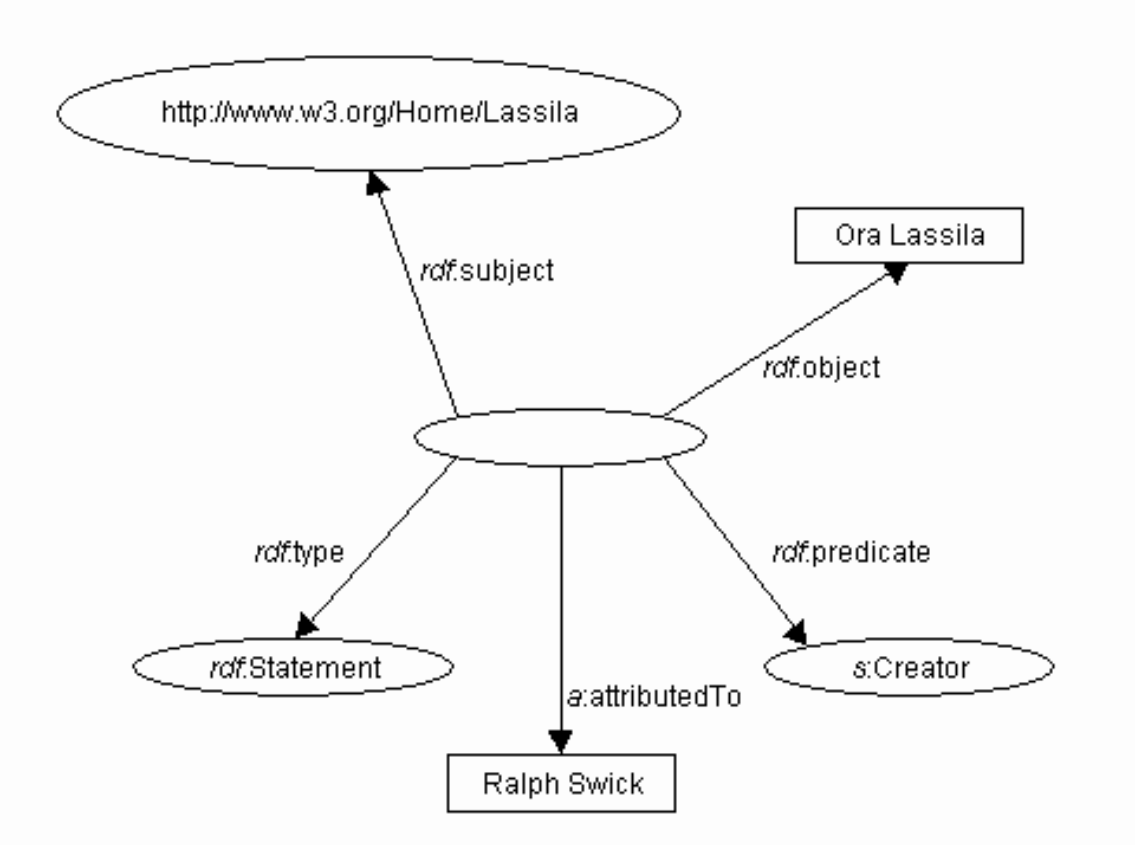

Figura 3.6: Esempio di reified statement RDF.

RDF Schema

Come già accennato è possibile usare RDF per descrivere schema attraverso le primitive definite in [7]. Di seguito illustriamo le principali e il loro uso:

rdfs:Resource è la classe che raggruppa tutti i soggetti di descrizioni RDF, i quali sono implicitamente istanze di questa classe;

rdfs:Class tutte le classi sono istanze della classe rdfs:Class, essa corrisponde al generico concetto di *tipo* o *categoria*;

rdf:Property alcune risorse RDF sono property, vengono usate come predicate in uno statement e sono istanze di questa classe;

rdfs:domain questa meta-property ha per subject una property e per ob-

ject una classe, indica a che tipo di oggetti è applicabile la property specificata;

rdfs:range la meta-property range indica di che tipo può essere il valore restituito da una property;

rdf:type questa property indica che una certa risorsa è membro di una certa classe, e possiede pertanto tutte le caratteristiche comuni agli elementi di quella classe. Essa ha domain rdf:Resource e range rdfs:Class;

rdfs:subClassOf indica che una classe A è sottoclasse di una classe B (B A), tutte le istanze di A saranno allora istanze di B (cioè una risorsa di tipo A è anche di tipo B). Come ci si aspetta questa property è transitiva, perciò (C B) ^ (B A) ! (C A);

rdfs:subPropertyOf indica che una property è sotto-property di un'altra e permette di costruire tassonomie di property: ad esempio (parente genitore) indicherebbe che un genitore è necessariamente anche un parente;

3.4.2 F-logic.

F-logic [22] è un linguaggio di programmazione che cerca di unire un modello di dati orientato agli oggetti ad un modello di programmazione funzionale basato sulla logica predicativa. Da un lato F-logic mette a disposizione gli strumenti tipici della programmazione OO per dichiarare gerarchie di classi, metodi e relative istanze, con il supporto dell'ereditarietà e del polimorfismo; dall'altro il modello di programmazione logica è orientato non alla descrizione di procedure ma alla formulazione di espressioni delle quali è possibile verificare la correttezza interrogando un database di oggetti.

Sintassi.

L'alfabeto di F-logic è costituito da:

un insieme di *costruttori* F ; sono i simboli funzionali[28] di F-logic, ad essi è associata una arietà, un numero intero non negativo che rappresenta il numero di argomenti richiesti dalla funzione;

un insieme infinito di *variabili* V;

i simboli ausiliari:

`(, ), [, ], ->, ->>, .->, .->>, =>, =>>,`

ecc;

i comuni connettivi logici: ^ ; _ ; : ; ; 8 ; 9 ;

Molecular formulas.

Una *molecola* in F-logic assume una delle seguenti forme:

1. Una asserzione *is-a* nella forma A :: B oppure C : B, dove A, B e C sono simboli funzionali o variabili o loro combinazioni. A :: B indica che la classe A è una sottoclasse della classe B, mentre C : B indica che l'oggetto C e un membro della classe B;

2. Una *object* molecola nella forma

   `O[`*`method-expression`*`].`

esistono inoltre vari tipi di *method expression*:

1. *scalar expression* non ereditabile (n 0)

   `ScalarMethod@`Q_1`,...,` Q_n `-> T`

2. *set-valued expression* non ereditabile (n; m 0)

   `SetMethod@`Q_1`,...,` Q_n `->> {`S_1`,...,`S_m`}`

3. *scalar expression* ereditabile (n 0)

   `ScalarMethod@`Q_1`,...,` Q_n `.-> T`

4. *set-valued expression* ereditabile (n; m 0)

   `SetMethod@`Q_1`,...,` Q_n `.->> {`S_1`,...,`S_m`}`

5. *scalar signature expression* ereditabile (n; m 0)

   `ScalarMethod@`Q_1`,...,` Q_n `=> (`A_1`,...,`A_m`)`

6. *set-valued signature expression* ereditabile (n; m 0)

```
SetMethod@Q1,..., Qn =>> (A1,...,Am)
```

Riprendendo gli esempi proposti nel par. 3.4.1, il modello illustrato in figura 3.4.1 può essere descritto con il seguente codice F-logic:

```
"http://www.w3.org/Home/Lassila" [
       creator@() -> "Ora Lassila"
       [2]
].
```

Il modello della figura 3.4.1 richiede la definizione di una classe "Bag":

```
Bag[
   li@(number) => (Student) [3]
].

Course[
   hasStudents@() => (Bag)
].

students:Bag [
   li@(1) -> "/Students/Amy":Student;
   [4]
   li@(2) -> "/Students/Tim":Student;
   li@(3) -> "/Students/John":Student;
   li@(4) -> "/Students/Mary":Student;
   li@(5) -> "/Students/Sue":Student;
].

"/courses/6.001":Course [
   hasStudents@() -> students
].
```

Rules e Queries

Una *rule* è uno statement nella forma

[2] La property "creator" è stata modellata con un metodo scalare non ereditabile (1).

[3] Definizione di signature (5)

[4] É possibile dichiarare il tipo di un oggetto nel momento stesso in cui lo si usa come in questo caso.

head body

dove *head* è una molecola e *body* è una espressione composta da molecole logicamente connesse. Ogni qual volta il body della rule è verificato, la head viene assunta come vera. Ad esempio il seguente codice:

```
FORALL X X:Student <-
   EXISTS W,
   Y:Course AND
   Y[hasStudents@()->Z] AND
   Z[li@(W)->X].
```

afferma che un oggetto appartiene alla classe "Student" a condizione che esso risulti iscritto ad un qualsiasi corso. Tutte gli esempi di codice visti finora sono rules composte esclusivamente dalla head, in cui il body è omesso in quanto sempre verificato.

Una query F-logic può essere considerata una rule particolare, senza head. Come per le rule, le variabili sono introdotte dai quantificatori `FORALL` e `EXISTS`: quando il body della query risulta verificato, le variabili sono istanziate.
Ad es. la query

```
FORALL X <- X:Course.
```

trova tutte le istanze della classe `Course` e le assegna alla variabile X .

Struttura di un programma F-logic.

Un programma F-logic consiste di tre parti:

> Una tassonomia di classi, anche detta *IS-A hierarchy*, le head delle rules che compongono questa parte conterranno esclusivamente dichiarazioni di tipo (A : B) e di sottoclasse (C :: B);
>
> Un'insieme di dichiarazioni di *signature*, le head di queste rules saranno composte solo da espressioni di signature;
>
> Una database di oggetti, le head di queste rules non conterranno alcuna espressione di tipo, sottoclasse o signature;

Tali restrizioni valgono solo per le head, mentre non esistono per il body, sarà dunque perfettamente lecito inserire, ad esempio, delle signature, nel body di una rule che dichiara un tipo, ad esempio:

```
FORALL X X:Merchandise <-
  (EXISTS W, X[price -> W]) OR
  (X:Y AND Y[price => ()]).
```

indica che un oggetto apparterrà alla classe "Merchandise" purché abbia un metodo "price" o sia istanza di una classe per cui esista una signature "price".

Esempio di Programma F-logic

```
/* class hierarchy */
Manager::Employee.
AreaManager::Manager.
TopManager::Manager.

/* type declarations */
max:Employee.
sally:Employee.
john:AreaManager.
lucy:TopManager.
jane:TopManager.
aProject:Project.
anotherProject:Project.

/* signatures */
Employee[ worksFor =>> Project ].
TopManager [ manages => Project ].

/* objects */
max  [ worksFor ->> {aProject, anotherProject} ].
sally[ worksFor ->> {aProject} ].
lucy [ manages -> {aProject} ].
jane [ manages -> {anotherProject} ].

/* rules */
FORALL X,Y
    X[ isResponsibleFor ->> Y ] <-
        EXISTS Z
        Y[ worksFor ->> Z ] AND
        X[ manages  ->  Z ].
```

Alcuni esempi di query che possono essere eseguiti sul database:

Trova tutti gli impiegati che lavorano ad un certo progetto:

```
FORALL X <-
    X [ worksFor ->> aProject ].

/* result */
X = max.
X = sally.
```

Trova tutti i `Manager`

```
FORALL X <-
    X:Manager.

/* result */
X = john.
X = lucy.
X = jane.
```

Trova tutti gli impiegati che non lavorano ad alcun progetto:

```
FORALL X <-
    EXISTS Y X:Employee
    AND NOT X[worksFor ->> Y]
    AND NOT X[manages -> Y].

/* result */
X = john.
```

Per ogni impiegato, trova il responsabile dei progetti a cui lavora:

```
FORALL X,Y <-
    Y [isResponsibleFor ->> X].

/* result */
X = max.
Y = lucy.
```

```
X = max.
Y = jane.

X = sally.
Y = lucy.
```

Capitolo 4

Reti Bayesiane

Il problema della personalizzazione rimanda direttamente a quello del ragionamento in presenza di incertezza. Si immagini ad esempio di voler codificare il fatto che l'attitudine di un utente a frequentare un certo tipo di ristorante dipende tanto dall'età e dalle disponibilità finanziarie dell'utente, quanto dal genere di ristorante e dalla spesa media per un pasto. In linea di principio un possibile approccio consiste nel raccogliere delle statistiche sull'età e sul reddito dei frequentatori di vari ristoranti e assumere tali dati come indicativi del grado di fiducia che riponiamo nella possibilità che un dato ristorante sia gradito a un dato utente. Consideriamo dunque *Età*, *Reddito*, *spesa Media* e *Specialità*, come variabili aleatorie discrete; la distribuzione di probabilità congiunta $P(E;R;M;S)$ potrà essere assunta come indice di gradimento, una volta noti i valori particolari assunti dalle variabili. In altre parole, possiamo rappresentare gli stati del dominio di interesse attraverso un insieme di attributi, ognuno dei quali rappresenta un particolare aspetto del nostro universo. Ogni attributo potrà assumere un insieme finito di valori mutuamente esclusivi. La rappresentazione esplicita della distribuzione di probabilità congiunta di tale insieme di attributi è chiaramente intrattabile da un punto di vista del costo computazionale. Il numero di possibili stati del nostro universo cresce infatti esponenzialmente rispetto al numero di attributi che lo descrivono.
Le *reti bayesiane*[21] permettono di specificare un modello di probabilità locale per ogni attributo, riducendo la dipendenza stocastica di una variabile aleatoria ad un ristretto insieme di altre variabili. Attraverso una rete bayesiana è possibile frammentare la distribuzione di probabilità $P(E;R;M;S)$ in un certo numero di distribuzioni più semplici; ciò rende il problema computazionalmente trattabile. La rete bayesiana in figura 4.1 individua quattro distribuzioni di probabilità, una per ogni attributo, che complessivamente caratterizzano interamente il nostro dominio: $P_1 = P(E)$, $P_2 = P(RjE)$,

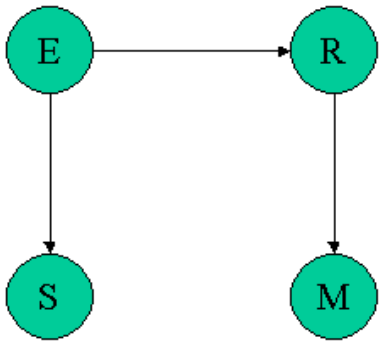

Figura 4.1: Esempio di rete bayesiana.

$P_3 = P(S|E)$, $P_4 = P(M|R)$, ognuna delle quali può essere rappresentata in modo esplicito senza troppa difficoltà. Dunque a partire dalla distribuzione di probabilità dell'età degli utenti, del reddito rispetto all'età, della spesa media di un ristorante rispetto al reddito dei suoi frequentatori, e della specialità offerta rispetto all'età dei frequentatori, possiamo, a partire dall'età e/o dal reddito di un utente valutare un indice di gradimento per un dato ristorante, assumendo come tale la probabilità che fra i suoi frequentatori si trovino individui con le caratteristiche dello specifico utente. È peraltro

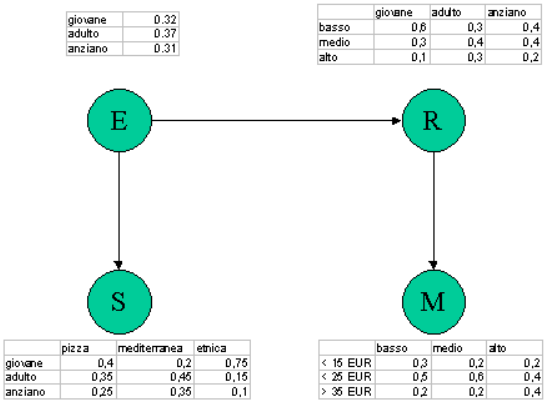

Figura 4.2: Esempio di rete bayesiana con distribuzioni di probabilità.

possibile applicare il ragionamento inverso e stimare la probabilità che un dato utente appartenga a una certa fascia di età o di reddito dato che ha

manifestato preferenza per certi ristoranti, la qual cosa si può dimostrare utile nella costruzione dinamica del profilo utente.

4.1 Reti Causali

Una rete causale è un grafo orientato in cui i nodi rappresentano eventi e gli archi rappresentano nessi causali tra i nodi. Una rete causale può essere usata per agevolare il ragionamento in presenza di incertezza, suggerendo le possibili conseguenze o le possibili cause di un evento.

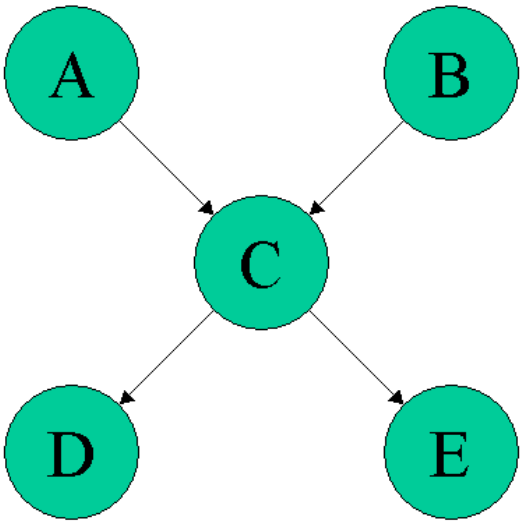

Figura 4.3: Una Rete Causale.

La rete causale in fig. 4.3 è costituita da cinque variabili e relativi nessi causali: gli eventi A e B hanno un impatto causale sull'evento C che, a sua volta, influenza gli eventi D ed E. Ogni nodo della rete è associato ad una variabile il cui stato possiamo osservare in un certo momento o desideriamo prevedere in un tempo futuro. Evidentemente in una rete di grandi dimensioni le dipendenze causali possono diventare troppo complesse o lunghe per essere gestite; la *d-separazione* è una proprietà di questo tipo di reti che permette di aggirare tale problema.

4.1.1 d-separazione

In particolari circostanze, osservato lo stato di una variabile, possiamo eliminare dal modello intere parti della rete causale, che non hanno più modo di influenzare le variabili alle quali siamo interessati. Una variabile il cui stato sia certo è detta *istanziata* e lo stato della variabile è chiamato *evidenza*.

Connessioni Seriali

Sempre in riferimento alla fig. 4.3, la sotto-rete {A,C,E} presenta una *connessione seriale tra le variabili*. In questo caso, osservare lo stato di A varierà la nostra fiducia nel possibile valore di C e, di conseguenza, di E. Analogamente, osservando il valore di E, modificheremo la nostra fiducia sul possibile stato di C e di A. Se però osserviamo lo stato di C, allora ogni possibile osservazione fatta su A non avrà più modo di influenzare la nostra fiducia in E e viceversa. Le variabili A ed E si dicono allora *d-separate dato C*
Ad es. sappiamo che la pressione atmosferica influenza la possibilità di pioggia e che la pioggia aumenta il rischio di incidenti stradali; possiamo affermare che in caso di bassa pressione il rischio di incidenti stradali è più alto, ma se osserviamo che sta, oppure non sta effettivamente piovendo, la pressione atmosferica diventa ininfluente.

Connessioni Divergenti

La sotto-rete {C, D, E} presenta una *connessione divergente*: in questo caso le variabili D ed E possono influenzarsi reciprocamente, se e solo se la variabile C non è osservata. Se ciò avviene il valore osservato di uno dei suoi figli non è più in grado di influenzare la nostra fiducia nello stato degli altri. Le variabili D ed E saranno allora d-separate dato C. Es.: la pioggia aumenta il rischio di incidenti stradali e di black-out, ma entrambi questi eventi hanno anche altre cause indipendenti dalla pioggia; se si verifica un black-out aumenta la nostra fiducia sulla possibilità di pioggia e di conseguenza sul rischio di incidenti. Se però osserviamo lo stato reale della variabile *pioggia* la comunicazione tra le variabili da essa dipendenti sarà bloccata.

Connessioni Convergenti

La sotto-rete A, B, C è caratterizzata da una *connessione convergente*. In questo caso le variabili sono d-separate se e solo se la variabile C *non* è istanziata. Es.: sappiamo che l'emicrania può essere causata da una malattia (ad esempio l'influenza) o dallo stress. Se le cause sono indipendenti, sapere che un persona è o non è malata non varia la nostra fiducia nella possibilità che

essa possa o meno soffrire di stress. Se tuttavia osserviamo che la persona soffre di emicrania e non ha alcuna malattia, aumenta la nostra fiducia nella causa alternativa.

Riassumendo, due variabili A e B sono d-separate se, per qualsiasi percorso possibile da A a B, esiste una terza variabile C tale che:

La connessione è seriale e C è istanziata;

oppure

La connessione è divergente e C è istanziata;

oppure

La connessione è convergente e nè C nè alcun suo discendente sono istanziate;

Per una data variabile A, l'insieme di variabili costituito dai predecessori di A, i suoi discendenti, e le variabili che condividono un discendente con A, è chiamato *blanket di Markov*. Se tutte le variabili del blanket di Markov per una variabile A sono istanziate, la variabile A è completamente d-separata dalla rete.

4.2 Definizione di Rete Bayesiana

Una rete bayesiana è un grafo orientato aciclico con le seguenti caratteristiche:

ogni nodo rappresenta un variabile aleatoria discreta con un numero finito di stati mutuamente esclusivi;

ad ogni variabile A con predecessori B_1, ..., B_n, è associata una distribuzione di probabilità $P(A|B_1, ..., B_n)$

Non è necessario che gli archi della rete bayesiana siano orientati secondo una direzione causa-effetto, ma è indispensabile che, in presenza di connessioni seriali, divergenti o convergenti, la d-separazione implicita nel modello rifletta la dipendenza tra le variabili esistente nel mondo reale. Nella rete in figura 4.4 la direzione della freccia non implica che la febbre sia causa dell'influenza. Il modello, per come è strutturato, richiede la specifica delle distribuzioni di probabilità P(FEBBRE) e P(INFLUENZA jFEBBRE) che sono più conformi ai tipici metodi di diagnosi: osservazione dei sintomi,

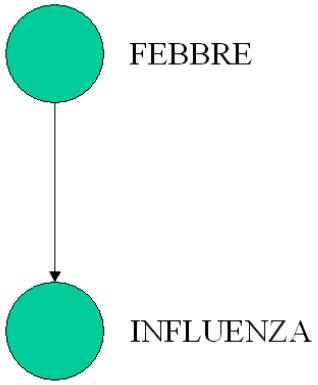

Figura 4.4: Una rete bayesiana per la dipendenza `febbre-influenza`.

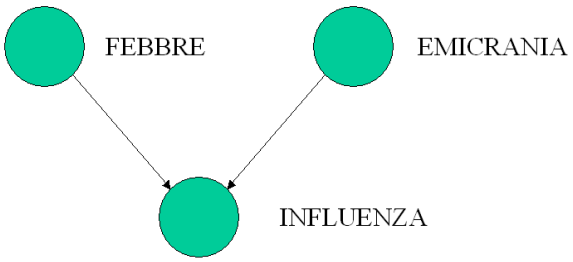

Figura 4.5: Una rete bayesiana per la dipendenza `febbre-influenza`, `emicrania-influenza`.

esame della frequenza dei sintomi nelle varie malattie, formulazione della diagnosi. Il modello in figura 4.5 introduce un altro predecessore per la variabile INFLUENZA . Ora sarà necessario specificare le distribuzioni di probabilità per P(FEBBRE), P(EMICRANIA) e
P(INFLUENZA jFEBBRE;EMICRANIA). Tuttavia il modello presenta un problema: le variabili FEBBRE ed EMICRANIA sono indipendenti, mentre non lo sono i fenomeni reali che esse rappresentano. Il modello effettuera delle predizioni non in linea con i dati sperimentali perché non tiene conto del contributo che la febbre apporta alla probabilità di emicranie. Tale contributo può essere modellato con la distribuzione
P(EMICRANIA jFEBBRE) come mostrato in figura 4.6.

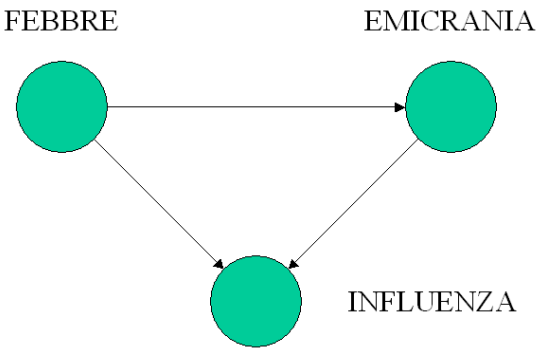

Figura 4.6: Una rete bayesiana per la dipendenza `febbre-influenza`, `emicrania-influenza` con la specificazione della dipendenza `febbre-emicrania`.

4.3 Design di Reti Bayesiane

Il primo passo nella costruzione di una rete bayesiana consiste nell'identificazione delle variabili che ne formeranno i nodi, esse sono di due tipi: variabili d'ipotesi e variabili d'informazione.
Le variabili d'ipotesi sono le variabili al cui stato siamo interessati e per le quali stiamo costruendo il modello. Le variabili d'informazione apportano suggerimenti sullo stato possibile delle variabili d'ipotesi; partendo da queste e sfruttando le inferenze che possiamo fare in base alla struttura del modello, è possibile stimare la probabilità che una variabile d'ipotesi assuma uno dei suoi possibili stati.
In questa fase preliminare si identificano le dipendenze tra le variabili sotto forma di archi non orientati. La determinazione dell'orientamento degli archi è spesso evidente, altre volte dipende dalla disponibilità di dati statistici. Qualora questi non siano disponibili la struttura della rete potrà essere semplificata.
È anche possibile acquisire in modo automatico la struttura della rete tramite un procedimento di apprendimento da ampi database di casi.
Una volta modellata la struttura della rete è necessario definire le distribuzioni di probabilità relative ad ogni nodo. Tale operazione può essere spesso condotta a partire da dati sperimentali e frequentemente la struttura della rete deve essere adattata alla disponibilità di dati statistici. Casi in cui una variabile abbia un numero eccessivo di predecessori o in cui non si riesca a determinare la Distribuzione di probabilità condizionata per una variabile richiederanno tali adattamenti, introducendo variabili intermedie o eventi

fittizi. Si faccia riferimento a [21] per una discussione dettagliata su tali problematiche.

Capitolo 5

Servizi Personalizzati

Il problema della personalizzazione può essere affrontato tanto con un approccio di tipo logico, basato sull'uso di ontologie, quanto con un approccio probabilistico. Tuttavia entrambe queste strategie presentano dei vantaggi e delle difficoltà.
I sistemi basati sulla classificazione dei contenuti possono sfruttare e generalizzare il feedback dell'utente: se l'utente esprime una preferenza, anche implicitamente soffermandosi su un dato documento, è possibile registrare la preferenza per tale risorsa e per la classe di cui essa fa parte. Tuttavia ciò richiede la stesura di regole specifiche e spesso complesse dato che una risorsa può essere classificata in molti modi, alcuni dei quali potrebbero non essere rilevanti ai fini della personalizzazione. Ad esempio se l'utente di un servizio di libreria virtuale perfeziona l'acquisto di un libro, alcune caratteristiche del prodotto (autore, genere...) possono essere usate per costruire un profilo utente, altre potrebbero essere casuali (anno di pubblicazione, prezzo, numero di pagine...), altre ancora quasi certamente sono ininfluenti ai fini della personalizzazione (codice ISBN, curatore della traduzione...).
Un modello probabilistico, come una rete bayesiana, permette invece di catturare l'incertezza di certe correlazioni sotto forma di nessi causali più o meno forti. Il feedback dell'utente può essere gestito efficacemente alterando la rete in modo tale che le funzioni di probabilità riflettano le frequenze relative delle scelte osservate.
Anche questo approccio presenta tuttavia dei limiti:

> la conoscenza incorporata in una rete bayesiana non sarà direttamente applicabile ad altri domini, anche se molto simili;
>
> è molto difficile mantenere una rete bayesiana in evoluzione, aggiungendo nuovi stati alle variabili o nuove variabili;

non c'è modo di ricondurre ai casi trattati nella rete un evento che non vi sia stato esplicitamente incluso;

I limiti delle due soluzioni proposte possono essere aggirati adottando un approccio ibrido. Le reti bayesiane orientate agli oggetti sono state proposte in [23] e [24] per semplificare il design e la manutenzione di reti bayesiane e renderle più flessibili applicando ad esse i principi del design object oriented. Incorporando in una ontologia informazioni di tipo probabilistico circa i nessi causali esistenti fra alcune delle properties in essa definite, otteniamo un rete bayesiana sulla quale è possibile, all'occorrenza, effettuare inferenze logiche. Il sistema per la personalizzazione di risorse e servizi descritto nel presente lavoro è stato realizzato come modulo di personalizzazione nell'ambito del progetto e-MATE [14],[13],[30], [31].

5.1 Ontologia Probabilistica

Si consideri ad esempio l'ontologia in fig. 5.2 e la relativa rete bayesiana in fig. 5.3. Esse esprimono un modello logico/probabilistico estremamente semplificato del dominio di un book shop online.

Nell'ontologia sono descritti i concetti e le relazioni che caratterizzano il dominio di interesse, in questo caso libri, autori ecc., e il modello dello User Profile, con tutte le risorse utili ai fini della personalizzazione. Alcune relazioni vengono inserite in una rete bayesiana che permette di associare all'ontologia un modello probabilistico. Ad esempio la relazione `genere` illustrata in figura 5.2 ha domain `Romanzo`, cioè si applica a istanze della classe `Romanzo` e range `Genere`, dunque il tipo degli elementi restituiti sarà `genere`. Ad esempio, in notazione F-Logic

```
Romanzo[ genere => Genere].
moby_dick:Romanzo.
avventura:Genere.
moby_dick [ genere -> avventura ].
```

Associamo alla relazione `genere` una distribuzione di probabilità che dia conto della diversa distribuzione dei libri venduti fra i vari generi letterari. Il numero di generi letterari è sicuramente finito e la distribuzione di probabilità è pertanto una distribuzione discreta che associa una probabilità ad ogni possibile valore della relazione `genere`; tutti i valori sono inclusi nell'ontologia. Similmente, ad altre relazioni dell'ontologia è associata la distribuzione di probabilità che le caratterizza. Alcune relazioni, come quelle illustrate in

figura 5.3 sono legate fra loro da nessi causali per indicare la dipendenza stocastica esistente fra di esse. In questo caso alla relazione *figlia* (nell'esempio `genere` e `autore`) è associata una distribuzione di probabilità condizionata. Per stimare l'indice di gradimento di un titolo in rapporto ad un dato utente il sistema esplora dapprima l'ontologia per individuare, fra le proprietà della risorsa stessa (cioè fra le proprietà che hanno come *domain* la classe di cui la risorsa è istanza o una sua super-classe), quali siano rilevanti ai fini della personalizzazione, cioè quali abbiano associata una distribuzione di probabilità; Il modello probabilistico può a questo punto essere impostato con i dati effettivi del profilo utente. Nella figura 5.3 sono mostrate tre relazioni legate da nessi causali, `romanzo.genere` e `libro.autore` hanno una distribuzione di probabilità che dipende da `persona.età`. Quest'ultima invece ha una distribuzione che rispecchia solo la reale distribuzione dell'età degli utenti. Senza ulteriori informazioni tale frammento di rete permette di calcolare le statistiche di vendita degli autori e dei generi letterari. Ma se la variabile `persona.età` viene istanziata all'effettivo valore dell'età dell'utente, il sistema è in grado di stimare una statistica di vendita corretta in funzione del profile utente.

Le preferenze espresse dall'utente, ad esempio acquistando un libro o suggerendolo ad altri utenti, forniscono un triplice feedback:

> sono inserite in un database di preferenze per essere utilizzate in futuro, ad esempio in caso di introduzione di nuovi nessi causali legati alle proprietà della risorsa: esse costituiscono il profilo transazionale dell'utente;
>
> sono usate per correggere il modello probabilistico specifico dell'utente rafforzando la probabilità associata allo specifico autore e genere letterario, tale approccio è analogo a quello dei sistemi *content-based*;
>
> sono usate per correggere il modello probabilistico di riferimento per tutti gli utenti, rafforzando le probabilità condizionali di autore e genere letterario, per profili utenti simili, attuando così una strategia di tipo *collaborative-filtering*;

Il modello logico fornito dall'ontologia permette di raggiungere un accordo sulla semantica delle variabili della rete bayesiana e sulla classificazione delle risorse, e di riportare i casi concreti alle classi generali previste nell'ontologia. Inoltre facilita l'applicazione della rete bayesiana a nuovi domini. L'ontologia probabilistica per il dominio della libreria online può infatti essere adattata, ad esempio, al dominio del cinema o dell'home video, estendendo opportunamente l'ontologia. Se il modello così esteso usa la stessa property, ad esempio,

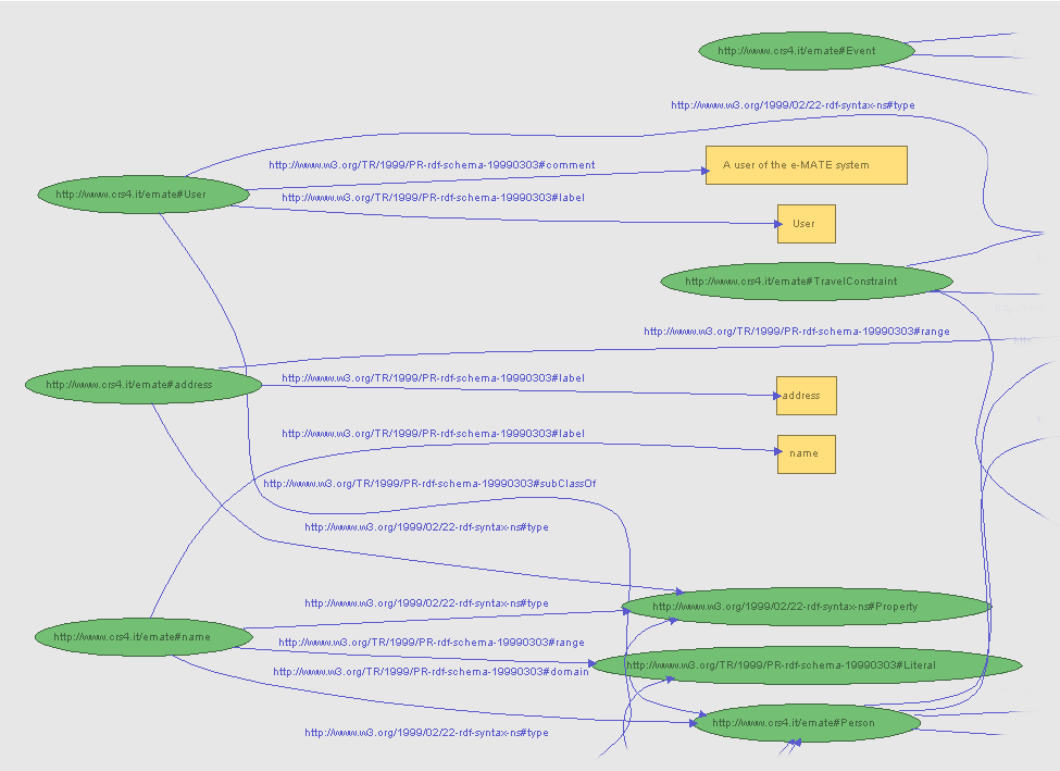

Figura 5.1: Un frammento dell'ontologia usata nel sistema e-MATE.

per `romanzo.genere` e per `film.genere`, o se la funzione di probabilità è associata ad una super-property comune, la conoscenza acquisita in un dominio può essere riutilizzata direttamente nell'altro.

5.2 Inferenze Logiche e Probabilistiche

Il vantaggio dato dall'uso di una ontologia in luogo di un comune database consiste nella possibilità di eseguire query complesse che richiedono inferenze basate sulla semantica delle proprietà e dei concetti, deducendo nuovi fatti a partire dai fatti noti.
Uno dei ragionamenti più basilari, ma non per questo banale, che l'uso di una ontologia rende possibile è quello relativo alle relazioni di tipo, sottoclasse, e sub-property. In particolare la transitività di queste due ultime relazioni permette di estendere un ragionamento risultato valido per un dato concetto o property a tutti i sotto-concetti. Si consideri il seguente esempio: avendo definito un criterio di preferenza per il concetto `Romanzo`, esso risulta valido

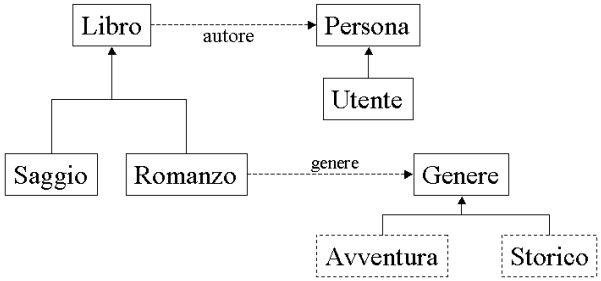

Figura 5.2: Esempio di ontologia per la personalizzazione di servizi.

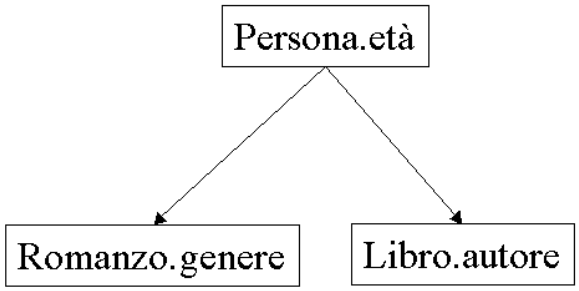

Figura 5.3: Esempio di rete bayesiana sulle properties dell'ontologia in fig. 5.2.

anche per tutte le sue sottoclassi.
Tale ragionamento può essere esteso alle relazioni

```
OperaNarrativa [
    ambientazione => Year;
    creatore => Persona;
    genereNarrativo =>> GenereNarrativo].

Romanzo::OperaNarrativa[
    autore::creatore => Persona].
    genereLetterario =>> GenereLetterario::GenereNarrativo;

Film::OperaNarrativa[
    trattoDa => OperaNarrativa;
    regista => Persona;
    sceneggiatore::creatore => Persona].
```

```
    genereCinematografico =>> GenereCinematografico::GenereNarrativo;

operaDrammatica:GenereNarrativo.
operaComica:GenereNarrativo.
commediaCinematografica::operaComica.
drammaCinematografico::operaDrammatica.
```

Le classi `Romanzo` e `Film` sono sottoclassi di `OperaNarrativa`, ed ereditano da questa la semantica delle relazioni `creatore` e `genereNarrativo` specializzandole rispettivamente in `autore`, `sceneggiatore`, `genereLetterario` e `genereCinematografico`.
Ogni fatto noto per la super-classe o le super-property sarà necessariamente valido per le sotto-classi e sotto-property. Esprimere una preferenza per il `genereCinematografico` chiamato `commediaCinematografica` comporta implicitamente una analoga preferenza per `operaComica`.

L'applicazione della transitività della relazione tassonomica non è l'unica strategia di inferenza possibile con le ontologie; relazioni inverse, transitive, simmetriche, riflessive, identita o disgiunzione tra classi ecc., permettono di determinare nuovi fatti a partire dai fatti noti. Nel caso specifico della personalizzazione ciò significa generalizzare le informazioni sulle scelte passate dell'utente, individuare le analogie tra di esse e usare tali informazione per prevedere le scelte future. Tali analogie non sempre sono ovvie, ad es. se un utente acquistasse i due romanzi: "The Remains of the Day" e "Howards End" potremmo giustamente dedurre che egli sia interessato a romanzi ambientati nell'Inghilterra della prima metà del XX secolo, ma un esame più attento, supportato da un'ontologia opportunamente dettagliata mostrerebbe che le riduzioni cinematografiche di tali romanzi sono state curate dallo stesso regista. Ciò potrebbe essere un caso oppure potrebbe essere il vero filo conduttore da sfruttare nel suggerimento della prossima scelta. Se invece trovassimo che le versioni cinematografiche sono state realizzate dallo stesso produttore, probabilmente saremmo spinti a considerare tale dato come una coincidenza. Tale questione richiede considerazioni di tipo probabilistico su quali proprietà di un oggetto siano solitamente considerate rilevanti al fine di compiere una scelta e come esse si rapportino tra loro e alle proprietà che caratterizzano lo specifico utente.

La rete bayesiana che si sovrappone all'ontologia la arricchisce di informazioni sulla dipendenza stocastica fra i valori assunti dalle property. L'esistenza o meno di tale dipendenza si può dedurre dall'esame di un database

di casi, ad esempio una statistica di vendite sufficientemente ampia. Una volta accertata l'esistenza di una dipendenza (ad es. tra `età` dell'utente e `genereCinematografico` o tra `professione` e `genereLetterario`), si definiscono le distribuzioni di probabilità relative alle property individuate. Poiché le variabili della rete bayesiana devono essere discrete, e poiché la probabilità di ogni valore che la variabile può assumere deve essere nota a priori, tutti i possibili valori devono essere inseriti nell'ontologia. Ciò comporta fra l'altro la necessità di definire degli intervalli per variabili continue come `età` o `reddito`.

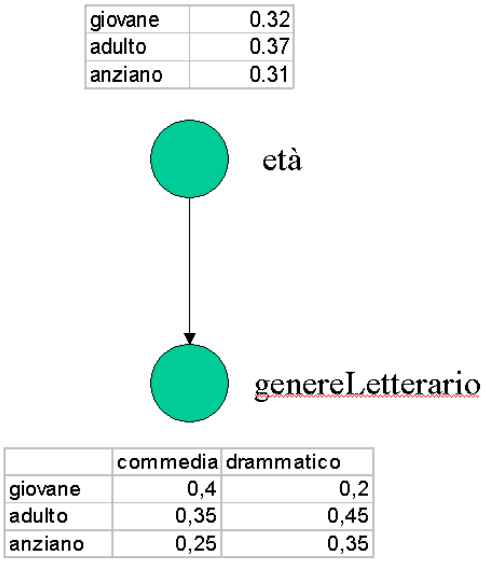

Figura 5.4: Tabella di probabilita condizionata per le variabili `età` e `genereCinematografico`.

Le scelte compiute dall'utente sono accuratamente registrate nel suo profilo transazionale ed usate per correggere le tabelle di probabilità della rete bayesiana: ogni volta che un utente `giovane` sceglie una `commedia` la probabilità

$$P(\texttt{genereCinematografico=commedia}|\texttt{età=giovane})$$

viene incrementata (l'incremento può essere fisso oppure può riflettere la frequenza relativa osservata) e la tabella viene normalizzata.
Se la rete bayesiana è condivisa da più utenti ciò equivale ad applicare una strategia *collaborative filtering*: poiché un utente giovane ha espresso una

preferenza per un certo genere cinematografico, esso risulterà rinforzato per tutti gli utenti giovani.
Il sistema di personalizzazione tenta dunque di prevedere le preferenze di un utente calcolando la probabilità di ogni property relativa agli oggetti fra cui l'utente è chiamato a scegliere: la valutazione di un film ad esempio produrrà un valore di probabilità per ogni property della classe `Film` (`genereCinematografico`, `regista`, `attoreProtagonista`, ecc.), i valori ottenuti sono poi combinati in un giudizio complessivo con una operazione di media. Per rendere conto del fatto che alcune property sono più influenti di altre nella scelta di un film la media può essere sostituita da una media pesata; il peso delle property può essere a propria volta inserito nell'ontologia. Si noti che non è necessario avere una distribuzione di probabilità direttamente associata alle property dell'oggetto sotto esame, infatti poiché

`genereNarrativo` `genereCinematografico`

ogni caratteristica del primo *deve* essere ereditata dal secondo, incluse le dipendenze probabilistiche. In tal caso però il feedback dell'utente non potrà essere inserito direttamente nella rete dato che esso non è relativo alla property che possiede la distribuzione di probabilità ma ad una sub-property che ne estende e specializza la semantica. In caso contrario ogni preferenza espressa per `genereCinematografico` verrebbe automaticamente estesa a `genereLetterario`; ciò potrebbe sembrare ragionevole a prima vista ma se l'ontologia riporta come distinte (anche se legate da una super-property comune) le due property, è perché si vuol mettere in risalto la differenza esistente fra di esse.
Analogamente a quanto detto per le property, è possibile risalire la tassonomia delle classi alla ricerca di property utili per la valutazione di un oggetto. In questo caso però se la property viene ereditata senza apportare variazioni semantiche (ad es. `OperaNarrativa.ambientazione` è ereditata dalle sottoclassi senza che queste la ridefiniscano o specializzino in alcun modo) la sua distribuzione di probabilità può essere alterata in funzione del feedback dell'utente. In questo caso si ha un funzionamento di tipo content-based: avendo registrato una preferenza per un film ambientato nel XIX sec. il sistema raccomanderà con maggiore enfasi film (e romanzi) ambientati in tale periodo.

Capitolo 6

Conclusioni

Nè le ontologie nè le reti bayesiane, applicate al problema della personalizzazione, sono in grado, separatamente, di offrire un modello tanto strutturato e al tempo stesso tanto flessibile, da rappresentare i vincoli e le caratteristiche dei domini reali.
Le ontologie permettono di creare modelli estremamente strutturati, dalla semantica ben definita, che siano di supporto alla comunicazione tra sistemi software (ad esempio client e server Web); tuttavia le dipendenze tra vari concetti devono essere rappresentate sotto forma di assiomi (regole) la cui stesura risulta spesso complessa e onerosa, e che comunque non sempre sono in grado di catturare l'inerente incertezza di alcune relazioni.
Le reti bayesiane permettono di codificare tale incertezza ma non posseggono le caratteristiche di generalità che sono invece proprie delle ontologie. In particolare la mancanza di struttura e la semantica implicita le rendono inadatte all'uso in domini applicativi estremamente dinamici, come i Web services (o i futuri Semantic-Web services), dove la flessibilità e l'interoperabilità rappresentano caratteristiche irrinunciabili.

Il principale problema dato dall'uso di ontologie e reti bayesiane consiste nella complessità dei modelli coinvolti: le ontologie sono uno strumento di recente invenzione e molti aspetti del loro uso (integrazione, apprendimento...) sono tuttora oggetto di ricerca di base; le reti bayesiane sono basate su una teoria matematica consolidata ma vari aspetti, come l'uso di distribuzioni continue o l'apprendimento automatico della struttura e delle probabilità dai dati, sono tuttora aperti. Inoltre la definizione dell'ontologia e della rete bayesiana sono operazioni generalmente abbastanza complesse, che anche per domini relativamente ristretti (poche decine di concetti) richiede l'intervento di esperti del dominio, di esperti in knowledge engineering, di esperti nel design di reti bayesiane e di un vasto database di casi per la determinazione

delle distribuzioni di probabilità. Questi problemi rendono difficoltosa l'applicazione a domini non banali delle idee presentate.

L'accoppiamento di ontologie e reti bayesiane rappresenta tuttavia un approccio alla personalizzazione ricco di potenzialità perché permette di codificare tramite la rete bayesiana quelle dipendenze tra i concetti, spesso considerate dominio del "senso comune", che l'ontologia non è in grado di catturare. Tale approccio risulta efficace in tutti i contesti in cui la necessità di condurre inferenze sotto incertezza suggerirebbe l'uso di un modello probabilistico, e contemporaneamente si desideri adottare un modello molto strutturato del dominio di interesse per definire chiaramente la semantica dei concetti usati, supportare l'interoperabilità tra diversi componenti software, e condurre inferenze logiche sulla base del modello. Sempre grazie all'ontologia sarà massimizzata la riusabilità del software e dei dati raccolti, sia per integrare nuovi concetti nel modello, sia per adattare il modello a nuovi domini.
Ontologie e reti bayesiane possono inoltre essere integrate in modo semplice e intuitivo: tanto l'ontologia quanto la rete bayesiana sono rappresentate da un grafo orientato (aciclico nel caso di quest'ultima) e tale omogeneità permette di considerare la rete bayesiana come un sotto-insieme dell'ontologia, costituito da meta-properties che legano una distribuzione di probabilità ad alcune relazioni.
Tali meta-properties sono parte integrante dell'ontologia e, insieme alle altre properties e i concetti in essa definiti, possono essere oggetto di query, inferenze, controlli di integrità ecc., rendendo il modello complessivo robusto e versatile.

Bibliografia

[1] Gediminas Adomavicius and Alexander Tuzhilin. Using data mining methods to build customer profiles. *IEEE Computer*, 34(2):74–82, 2001.

[2] Tim Berners-Lee. Semantic web roadmap, Sept. 1998.

[3] Nigel Bevan. Quality in use: Meeting user needs for quality. *Journal of System and Software*, 1999.

[4] Kurt Bollacker, Steve Lawrence, and C. Lee Giles. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the Second International Conference on Autonomous Agents*, pages 116–123, New York, 1998. ACM Press.

[5] Kurt Bollacker, Steve Lawrence, and C. Lee Giles. A system for automatic personalized tracking of scientific literature on the web. In *Digital Libraries 99 - The Fourth ACM Conference on Digital Libraries*, pages 105–113, New York, 1999. ACM Press.

[6] T. Bray, J. Paoli, and C. M. Sperberg-McQueen (Eds). "Extensible Markup Language (XML) 1.0 (2nd Edition)". W3C Recommendation, oct 2000. http://www.w3.org/TR/2000/REC-xml-20001006.

[7] Dan Brickley and R.V. Guha. Resource description framework (rdf) schema specification, 2000.

[8] Ibrahim Cingil, Asuman Dogac, and Ayca Azgin. A broader approach to personalization. *Communications of the ACM*, 43(8):136–141, 2000.

[9] K. Decker, M. Williamson, and K. Sycara. Matchmaking and brokering, 1996.

[10] Mariano Fernández, Asunción Gómez-Pérez, and Natalia Juristo. METHONTOLOGY: From ontological art towards ontological engineering.

In *AAAI-97 Spring Symposium on Ontological Engineering*, pages 33–40, 1997.

[11] Enrico Franconi. Description logics for conceptual design, information access and ontology integration, 2002. Tutorial at the First International Semantic Web Conference, 2002.

[12] C. Lee Giles, Kurt Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In Ian Witten, Rob Akscyn, and Frank M. Shipman III, editors, *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, June 23–26 1998. ACM Press.

[13] Sylvain Giroux, Claude Moulin, Gavino Paddeu, Davide Carboni, Roberto Demontis, Stefano Sanna, and Enrico Stara. Mobilite, personnalisation et information geo-referencee. In *Journées francophones pour l intelligence artificielle distribuée et les systèmes multi-agents*, 2001.

[14] Sylvain Giroux, Eloisa Vargiu, Claude Moulin, Stefano Sanna, Alessandro Soro, and Gavino Paddeu. e-mate: An open architecture to support mobility of users. In *Fifth International Baltic Conference on Databases and Information Systems*, 2002.

[15] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[16] Nicola Guarino and Christopher Welty. Identity and subsumption, 2001.

[17] Nicola Guarino and Christopher A. Welty. A formal ontology of properties. In *Knowledge Acquisition, Modeling and Management*, pages 97–112, 2000.

[18] Nicola Guarino and Christopher A. Welty. Ontological analysis of taxonomic relationships. In *International Conference on Conceptual Modeling / the Entity Relationship Approach*, pages 210–224, 2000.

[19] Haym Hirsh, Chumki Basu, and Brian D. Davison. Learning to personalize. *Communications of the ACM*, 43(8):102–106, 2000.

[20] Clyde W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Communications of the ACM*, 45(2):42–47, 2002.

[21] Finn V. Jensen. *Bayesian networks and decision graphs.* Statistics for engineering and information science. Springer, 2001.

[22] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. Technical Report TR-90-003, 1, 1990.

[23] Daphne Koller and Avi Pfeffer. Object-oriented bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 302–313, 1997.

[24] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *AAAI/IAAI*, pages 580–587, 1998.

[25] Paavo Kotinurmi. User profiles and their management. Technical report, HUT, Software Business and Engineering institute.

[26] Alexander Maedche. Development and applications of ontologies, 2001.

[27] Udi Manber, Ash Patel, and John Robison. Experience with personalization of yahoo! *Communications of the ACM*, 43(8):35–39, 2000.

[28] Vincenzo Manca. *Logica Matematica*. 2001.

[29] John McCarthy. Phenomenal data mining. *Communications of the ACM*, 43(8):75–79, 2000.

[30] Claude Moulin, Sylvain Giroux, Raffaella Sanna, and Alessandro Soro. Personnaliser des documents en composant des services. In *Documents Virtuels Personnalisables*, 2002.

[31] Claude Moulin, Sylvain Giroux, Raffaella Sanna, Alessandro Soro, and Gavino Paddeu. Using shared ontologies for communication and personalization. Technical report, CRS4, 2002.

[32] Alexandrin Popescul, Lyle Ungar, David Pennock, and Steve Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *17th Conference on Uncertainty in Artificial Intelligence*, Seattle, Washington, August 2–5 2001.

[33] Kirsten Swearingen Rashmi. Beyond algorithms: An hci perspective on recommender systems.

[34] G. Salton and C. Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, (29):351–372, 1973.

[35] J. Ben Schafer, Joseph A. Konstan, and John Riedi. Recommender systems in e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–166, 1999.

[36] York Sure, Michael Erdmann, Juergen Angele, Steffen Staab, Rudi Studer, and Dirk Wenke. Ontoedit: Collaborative ontology development for the semantic web. In Ian Horrocks James Hendler, editor, *Proceedings of the First International Semantic Web Conference, 2002*, pages 221–235, Berlin Heidelberg, 2002. Springer-Verlag.

[37] Ora Lassila Tim Berners Lee, James Hendler. The semantic web, 2001.

[38] Mike Uschold and Michael Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.