Politecnico di Milano

Facoltà di Ingegneria dell'Informazione

un'estensione temporale delle logiche descrittive fuzzy

---

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Tesi di laurea in Ingegneria Informatica V. O.
candidato: Marco Furlan
matricola 645420
anno accademico 2006 – 2007

Relatore: Chiar.mo Prof. Marco Colombetti

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

1

Introduzione

1.1 Obiettivo con motivazioni

In questa tesi ho voluto dare un contributo al *Semantic Web*. Mi sono proposto di esplorare la possibilità di estendere temporalmente le Logiche Descrittive o *Description Logics* (DL) crisp, di applicare l'estensione temporale alle DL *fuzzy* e di soffermarmi su quali nuove proprietà degne di interesse se ne possano estrarre.

Le DL sono una classe di linguaggi logici del primo ordine (FOL): sono formalismi progettati per la ricostruzione logica e l'estensione di strumenti di Rappresentazione della Conoscenza (*Knowledge Representation*). Oggi, le DL sono anche considerate il più importante formalismo unificante per i molti linguaggi di rappresentazione centrati sugli oggetti, usati in aree diverse da quella della Rappresentazione della Conoscenza (1). Le caratteristiche più importanti delle DL sono l'alta espressività assieme alla decidibilità, garanzia quest'ultima che gli algoritmi di *reasoning* terminino sempre con le risposte corrette.

Le DL sono e saranno impiegate nel *Semantic Web* (2), il web di terza generazione, che si propone di convertire la mera ricerca automatica d'*Informazione*, come è ora possibile, in ricerca automatica di *Conoscenza*, cioè ricerca intelligente dei contenuti. A questo fine, esso mira a descrivere i propri documenti multimediali tramite concetti e relazioni strutturati in *ontologie*, per le quali possa essere applicato il *reasoning* automatico.

Il *Semantic Web* è un grande e appassionante progetto in corso d'opera. Il *reasoning* automatico, che opererà sul web, si comporrà di settori, ove le catene d'inferenze logiche saranno lunghe e le questioni di complessità del calcolo quindi essenziali, e di settori in cui, invece, prevarrà la ricchezza semantica dei concetti ed il *reasoning* sarà essenzialmente valutativo. Il mio vuole essere un contributo a quest'ultimo settore.

Sono partito dalla considerazione che gran parte delle *ontologie* realizzate finora sono fotografie istantanee di una parte del mondo e si impone che esse siano valide eternamente. Per esse si utilizzano linguaggi logici statici, indipendenti dal tempo.

Siamo tutti ben consci, invece, di come la vita sia legata al tempo in modo essenziale. Gran parte delle questioni d'informazione e conoscenza non ci riguardano hic et nunc, ma sono relative ad

un'evoluzione temporale, globale oppure individuale. Non esiste settore che non offra esempi. Nel campo sanitario, i medici sono spesso più interessati all'andamento nel tempo dei fattori fisiologici (pressione, glicemia, temperatura ecc.), piuttosto che ai singoli valori istantanei. Nel campo ecoambientale, si discute di riscaldamento globale: interessa poco la singola temperatura della singola località, ma più l'evoluzione nel tempo delle temperature di un campione significativo di località. Nel campo del controllo automatico, esistono sistemi intrinsecamente dinamici. In un sistema preda – predatori, interessa sapere se il predatore si avvicina alla preda o, ancor più, se una determinata parte dei predatori si avvicina alla preda.

Il mio approccio, valido per le logiche *crisp*, si applica facilmente alle logiche *fuzzy* e in esse si arricchisce. Le logiche fuzzy sono un'estensione delle logiche tradizionali aristoteliche (dette anche *crisp*), quelle del vero-falso e terzo escluso, dove ogni elemento appartiene ad un concetto oppure non vi appartiene, cioè vi appartiene con valore 0 o 1, in modo esclusivo. Nelle logiche fuzzy sono permessi invece diversi gradi d'appartenenza, compresi tra 0 e 1, e quindi sono ammessi concetti dai contorni sfumati, assai vicini all'intendere umano, quali *Vicino*, *Giovane* ecc., e che hanno difficoltà ad essere espressi nelle logiche crisp.

Il mio intento è stato dunque ricercare un modo nuovo di esprimere alcune classi di concetti fuzzy contenenti il divenire, inteso come passaggio da un concetto ad un altro, quali *Caduto*, *Avvicinato*, *Riscaldato* ecc., e di ottenere altri concetti più complessi, che potessero esprimere gli andamenti di crescenza, decrescenza o costanza e le relative deviazioni od oscillazioni nell'appartenenza ad uno stesso concetto di base. Inoltre, ho voluto considerare i *quantificatori* fuzzy, che già esistono in alcune logiche fuzzy, e li ho estesi temporalmente, ottenendo i *quantificatori temporali*, che corrispondono agli avverbi di tempo nel linguaggio parlato: *Spesso*, *A volte*, *Solitamente, Circa2voltesu3* ecc.

In un sistema così descritto, numerose sono le query possibili a semantica del tutto originale.

1.2 Contributi originali

Esistono in letteratura alcune estensioni temporali di alcune DL; il mio approccio, tuttavia, ha caratteristiche originali. Nella maggior parte delle pubblicazioni, il tempo non entra a far parte interna dell'ontologia, ma è l'interpretazione, che è esterna all'ontologia e ne definisce la semantica, ad essere dipendente dal tempo. Inoltre, a mia conoscenza, non esiste un'estensione temporale fuzzy organicamente strutturata.

Il mio approccio mantiene la struttura di base delle DL statiche: in particolare, prende in considerazione le logiche appartenti alla sottoclasse cosiddetta $\mathcal{SHOIN}$(D). Queste logiche hanno la caratteristica di essere decidibili ed avere una buona espressività e sono utilizzate, tra gli altri, dal linguaggio OWL-DL, un linguaggio per la definizione delle ontologie. Introduco la dimensione temporale internamente al linguaggio logico, come un dominio concreto, costituito da istanti di tempo puntuali a valore reale, eventualmente, ma non necessariamente, intervallati da distanza costante. L'interpretazione può rimanere così indipendente dal tempo.

Ho definito un'Ontologia Superiore (*Upper Ontology*) che partiziona il mondo in individui statici, non soggetti ad evoluzione, ed individui dinamici, dipendenti dal tempo.

Ogni individuo dinamico viene descritto associandogli i nuovi concetti di Linea Universo *ULine* e di Evento *Event*, concetti mutuati dalla fisica moderna. Un individuo dinamico è rappresentato

univocamente da una e una sola Linea Universo. La Linea Universo di un individuo è costituita dalla sequenza dei suoi Eventi. Un Evento *Event* è la rappresentazione dell'individuo in un particolare istante temporale. L'istante temporale è associato ad ogni *Event* da un predicato concreto.

Gli *Event* corrispondono agli *eventi* spaziotemporali della fisica, ma potrebbero anche essere chiamati *realizzazioni*, *incarnazioni*, *manifestazioni, istanziazioni o avatar* della Linea Universo di un individuo. La Linea Universo *ULine*, composta di *Event*, a sua volta si potrebbe vedere anche, in se stessa, come *l'astrazione* o il *se'* dell'individuo.

Ogni *Event* è dunque caratterizzato dalla *ULine* cui appartiene e dal valore del dominio temporale ad esso associato. Mentre ogni *Event* appartiene ad una e una sola *ULine*, un istante temporale, ovviamente, può caratterizzare *Event* di *ULine* distinte: si hanno *Event* contemporanei, precedenti e successivi. Dall'ordinamento del dominio temporale viene dunque indotto un ordinamento degli *Event*.

Parlando delle ontologie particolari, esistono concetti che costituiscono proprietà immutabili dell'individuo. Ad esempio, il concetto di *Persona*: se l'individuo marco è *Persona*, lo è per tutta la sua esistenza; oppure il concetto di *Lampadina* per l'individuo $\mathrm{lampadina}_n$. Nelle ontologie, questi concetti sono attribuiti alla *ULine* dell'individuo e sono ereditati da ogni suo *Event*.

Altri concetti, invece, sono proprietà transeunti e modificabili dell'individuo. Ad esempio, il concetto di *Funzionante* (crisp) per un individuo $\mathrm{lampadina}_n$ o *Felice* (fuzzy) per l'individuo marco. Questi concetti sono attribuiti, nell'ontologia, ai singoli *Event* interessati e non riguardano la *ULine*.

Se osassi scomodare Aristotele, forse non sarebbe in totale disaccordo con questa parte della mia impostazione che riguarda la semantica dei concetti statici. Aristotele potrebbe dire che un concetto statico proprio della *ULine* è *Sostanza* (οὐσία), mentre un concetto statico proprio di un *Event*, ma non della sua *ULine*, è *Accidente* (συμβεβηκός) (3). Chiaramente, Aristotele aveva introdotto *Sostanza* ed *Accidente* in un universo di concetti crisp: egli è il padre, appunto, della logica aristotelica. Nel nostro caso, consideriamo, in generale, anche concetti fuzzy.

Oltre a questi concetti, che hanno una valutazione istantanea, esiste nel linguaggio parlato una vasta classe di concetti che hanno semantica, invece, intrinsecamente dinamica. Mi riferisco a concetti come *Arricchito*, *Invecchiato*, *Sollevato*, *Avvicinato* e ogni concetto sul divenire.

Per descrivere questi, definisco una nuova semantica per alcuni operatori temporali, già esistenti in letteratura, che, applicati a concetti, costruiscono nuovi *concetti dinamici*.

Essi sono gli operatori *sometime* $\Diamond$, *always* $\Box$, *Until* $\mathcal{U}$, *Since* $\mathcal{S}$, *Next* $\oplus$, *Prev* $\ominus$. I concetti dinamici richiedono, per essere valutati su un individuo, di considerare necessariamente tutto o una parte dell'insieme dei suoi *Event*.

sometime $\Diamond$ ed *always* $\Box$ sono operatori unari, *Until* $\mathcal{U}$ e *Since* $\mathcal{S}$ sono binari, *Next* $\oplus$ e *Prev* $\ominus$ sono anch'essi unari.

Con essi posso dire se la mia *Pressione* è stata "qualche volta *(sometime) Alta*" oppure è stata "sempre *(always) Normale*" oppure se è stata "*Alta* finché (*Until*) non è iniziata la terapia" oppure se è stata "*Normale* da quando (*Since*) è iniziata la terapia". Inoltre, posso valutare il valore della mia pressione nell'istante successivo *(Next)* o precedente *(Prev)* a quello in esame.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Tutto ciò vale sia per concetti crisp che per concetti fuzzy. Con i concetti fuzzy l'espressività del linguaggio aumenta (assieme, purtroppo, alla complessità).

Combinando la sussunzione tra concetti $\sqsubseteq$ con gli operatori $\oplus$ e $\ominus$, ho ottenuto due nuovi operatori che ho chiamato *SubsNext* e *SubsPrev*.

Dato un concetto C, il nuovo concetto *SubsNext(C)* è ottenuto da $C \sqsubseteq \oplus C$, cioè dalla sussunzione tra C e $\oplus C$, dove $\oplus C$ è il concetto contenente, per ogni *Event*, il valore in C assegnato all'*Event* successivo. Analogamente, *SubsPrev(C)* è ottenuto dalla sussunzione $C \sqsubseteq \ominus C$.

Sappiamo come la sussunzione tra due insiemi indichi l'inclusione o il grado di inclusione, nel caso fuzzy, di un insieme nell'altro.

Viene spontaneo allora osservare che, considerato un certo insieme di individui *Event*, qualora ci sia una crescenza nel tempo dei valori assunti da questi *Event* in un dato concetto C, il valore di ognuno di questi *Event* è $\leq$ al valore dell'*Event* successivo. Dunque, i valori di questi *Event* nel concetto C sono $\leq$ ai valori degli stessi *Event* nel concetto $\oplus C$.

Se gli *Event* considerati corrispondono ad uno stesso individuo *ULine* x, allora dire che, per gli *Event* di x, il valore in C cresce col tempo equivale a dire che, per gli *Event* di x, C è insiemisticamente incluso in $\oplus C$, ovvero che per x vale $C \sqsubseteq \oplus C$, cioè x appartiene al concetto *SubsNext(C)*, che equivale a dire che *SubsNext(C)*(x) = 1.0. L'appartenenza di una *ULine* al concetto *SubsNext(C)* corrisponde alla proprietà di crescenza del concetto C.

Questo vale per concetti crisp: ad esempio se l'individuo lampadina_n passa dallo stato di spento allo stato di acceso, il suo valore nel concetto *Acceso* passa da 0 a 1 e quindi *SubsNext(Acceso)*(lampadina_n) = 1.0.

Inoltre, vale per i concetti fuzzy: se l'individuo predatore_p passa dalla distanza dalla preda d1 alla distanza d2, con $d2 < d1$, il suo valore nel concetto *Vicino* passa da v1 a v2, con $v2 > v1$, e quindi *SubsNext(Vicino)*(predatore_p) = 1.0, supponendo l'avvicinamento monotòno.

L'operatore *SubsPrev* descrive in modo del tutto analogo la caratteristica inversa, cioè la decrescenza dei valori.

Cosa succede se l'andamento di crescenza (o decrescenza) non è monotòno? Nel caso di oscillazioni, la sussunzione crisp (inclusione perfetta) con concetti crisp dà valore 0 sia per *SubsNext* che per *SubsPrev*. Se invece consideriamo i casi fuzzy, cioè concetti fuzzy oppure sussunzione fuzzy oppure entrambi, si possono avere entrambi i valori non nulli.

Vale una caratteristica importante: in presenza di una crescenza in media, il valore di *SubsNext* è maggiore del valore di *SubsPrev* e viceversa in caso contrario. I due valori sono invece uguali in caso di costanza in media. La media è qui assunta come interpolazione lineare dei dati.

Inoltre, per ogni andamento in media, i valori dei concetti *SubsNext(C)* e *SubsPrev(C)* sono tanto inferiori quanto maggiori sono le ampiezze delle oscillazioni dei valori in C.

Date queste considerazioni, mi è stato naturale, dato il generico concetto C, definire la terna di nuovi concetti:

$\nearrow C \equiv IncrC \equiv Incr(C)$, $\searrow C \equiv DecrC \equiv Decr(C)$ e $\rightleftharpoons C \equiv ConstC \equiv Const(C)$, con la seguente semantica:

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$$\nearrow C^{\mathcal{I}}(x) \equiv IncrC^{\mathcal{I}}(x) = \max\{ SubsNextC^{\mathcal{I}}(x) - SubsPrevC^{\mathcal{I}}(x), 0.0\},$$

$$\searrow C^{\mathcal{I}}(x) \equiv DecrC^{\mathcal{I}}(x) = \max\{ SubsPrevC^{\mathcal{I}}(x) - SubsNextC^{\mathcal{I}}(x), 0.0\},$$

$$\rightleftharpoons C^{\mathcal{I}}(x) \equiv ConstC^{\mathcal{I}}(x) = \text{if } SubsNextC^{\mathcal{I}}(x) = SubsPrevC^{\mathcal{I}}(x) \text{ then } SubsNextC^{\mathcal{I}}(x) \text{ else } 0.0.$$

Questi concetti rappresentano, dunque, gli andamenti, rispettivamente crescenti, decrescenti e costanti, e la loro monotonia. Li ho chiamati *Concetti d'Andamento* (*Tendency Concepts*) e *operatori d'andamento* (*tendency operators*) gli operatori corrispondenti.

Ho fornito un'altra definizione per quelli che ho chiamato *Concetti d'Andamento Monotòno* (*Monotonic Tendency Concepts*), la cui semantica risente unicamente del grado di monotonia e non del valore dell'incremento o decremento.

Restando ai *Concetti d'Andamento*, se un individuo, nel tempo, ha visto il suo valore d'appartenenza al concetto C crescere in media, pur oscillando, egli sarà membro del concetto $\nearrow C$ e lo sarà con un grado tanto maggiore quanto maggiore è l'incremento e quanto inferiore è l'ampiezza delle oscillazioni. Analogamente per gli altri due concetti $\searrow C$ e $\rightleftharpoons C$. E' vero che una valutazione di crescenza è ottenibile anche con metodi statistici, quali l'interpolazione lineare, però quest'ultima dà la crescenza o decrescenza dell'andamento, non considera l'ampiezza delle oscillazioni. Altre statistiche lo fanno, ma allora sono piuttosto complesse. Il vantaggio principale, comunque, è che questi nuovi concetti si ottengono con semplici operazioni di sussunzione e sono concetti sui quali è possibile il *Reasoning* proprio delle Description Logics.

Utilizzando anche i quantificatori fuzzy (*Most*, *Some*, *Between*$(\sim 3)\&(\sim 5)$ ecc.), l'espressività aumenta ulteriormente. Ho introdotto una rappresentazione semplificata di essi con una generica membership function lineare a tratti di forma trapezoidale, che può degenerare in triangolo, spalla destra, spalla sinistra o rettangolo crisp.

In particolare, ho potuto, in modo naturale, con la sola applicazione di un ruolo, definire i nuovi *quantificatori temporali*, che corrispondono agli avverbi di tempo del linguaggio parlato associati alla frequenza, quali "*Solitamente*", "*Qualche volta*", "*Spesso*" ecc.

Ho dimostrato con due teoremi che i quantificatori temporali che si trovano alle estremità della scala della quantificazione e che potremmo chiamare "*Sometime*" ed "*Always*" equivalgono agli operatori temporali sopra menzionati, gli operatori *sometime* $\Diamond$ ed *always* $\Box$. Questo, a supporto della coerenza del sistema costruito.

Ho enunciato e dimostrato un terzo teorema, avente utilità pratica.

Quantificatori ed *operatori d'andamento* possono essere utili in numerose applicazioni: ad esempio, per valutare se effettivamente esiste il riscaldamento globale del pianeta, cioè se "la maggior parte delle località ha temperature (mediamente) crescenti":

$$GlobalHeating = (Most)hasLocality.\nearrow TempLocality\,.$$

Oppure, potremmo valutare, nel controllo automatico di un sistema prede – predatori, quali prede siano in pericolo, supponendo in pericolo le prede per cui almeno la metà dei predatori siano solitamente vicini:

$$UsuallyInDangerPray = (MoreThanHalf)isPrayFor.(Usually)ClosePredator$$

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

oppure, alternativamente, supponendo in pericolo le prede per cui almeno la metà dei predatori siano costantemente vicini:

$$ConstInDangerPray = (MoreThanHalf)isPrayFor.ConstantClosePredator$$

$$=(MoreThanHalf)isPrayFor.((Sometime)ClosePredator \sqcap \rightleftharpoons ClosePredator)$$

Le prede possono rappresentare obiettivi di diversa natura e dimensione, che per essere raggiunti necessitano di automi non umani, per motivi ambientali, di sicurezza o di dimensione. Si può pensare ad una chiazza di petrolio da dissolvere, ad una profondità sottomarina o una cavità vulcanica da esplorare, alle mine da eliminare, fino alle metastasi di un tumore nell'organismo umano da aggredire.

Un altro esempio, in campo sanitario. Possiamo valutare se l'andamento della pressione (o di qualsiasi altro indice fisiologico) si mantenga "ragionevolmente" costante nei valori normali.

Se *NormalPressure* è un concetto opportunamente definito, e

$NormalPressurePatient = Patient \sqcap \exists hasPressure.NormalPressure$, allora

$ConstNormalPatient = (Sometime)\ NormalPressurePatient \sqcap \rightleftharpoons NormalPressurePatient$

è il concetto che rappresenta i pazienti con i valori di pressione mediamente costanti entro la normalità.

Se, ad esempio, sono state introdotte alcune nuove terapie sperimentali e ci interessa sapere quali siano efficaci per la maggior parte dei pazienti, potremmo descrivere efficaci quelle terapie per cui la maggior parte dei pazienti che l'assumono ha una pressione che si mantiene costantemente nella norma:

$$EffectiveTherapy = (Most)takenBy.ConstNormalPressurePatient.$$

I costrutti utilizzati rendono questa un'estensione della classe delle DL $\mathcal{SHOIQ}^{+}$(D) fuzzy. Potremmo chiamarla $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D).

1.3 Schema dell'opera

Questo mio lavoro è articolato nei seguenti capitoli.

Nel capitolo 2 prendo in considerazione lo stato dell'arte riguardo ai principali argomenti coinvolti.

Nel paragrafo 2.1 offro una panoramica delle logiche DL fuzzy: espressività, problemi con le definizioni delle norme, sintassi e semantica di alcune classi di linguaggi logici. Definisco una *Knowledge Base* e introduco i problemi d'inferenza di base.

Nel paragrafo 2.2 introduco il concetto dei quantificatori fuzzy, assoluti e relativi, e descrivo il problema della cardinalità degli insiemi fuzzy. Concludo con il metodo *GD* (4) per le valutazioni delle espressioni quantificate.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Nel paragrafo 2.3, descrivo brevemente i contributi in letteratura alle logiche DL temporali e alle DL temporali fuzzy.

Nel capitolo 3, introduco i modelli matematici oggetto di questa tesi.

Nel paragrafo 3.1 descrivo il modello fuzzy $f\_\mathcal{ALCQ}^{+}$(D), che ammette i quantificatori, considerati come insiemi fuzzy, e dove le espressioni quantificate sono interpretate con il metodo *GD*. Introduco poi una funzione parametrizzata lineare a tratti *Trap*, di forma genericamente trapezoidale, che sarà utilizzata nella definizione delle membership functions dei concetti concreti e dei quantificatori. Espressioni quantificate valide saranno dei due tipi $QR.C$ e $Q\,C \sqsubseteq D$.

Il paragrafo 3.2 costituisce il cuore di questo lavoro, dove definisco il mio modello temporale. Il modello temporale è definito progressivamente, in un percorso di accrescimento dell'espressività. Inizio in 3.2.1, con le logiche crisp, introducendo una partizione dell'Universo e una struttura a semantica temporale che determina un ordinamento sugli individui. Definisco sintassi e semantica dei concetti dinamici (temporali) ottenuti con gli operatori temporali $\Box$, $\Diamond$, $\mathcal{U}$ e $\mathcal{S}$. Definisco gli operatori *Next* $\oplus$ e *Prev* $\ominus$ e alcune considerazioni sulla sussunzione mi portano alla definizione dei nuovi operatori *SubsNext* e *SubsPrev* e degli *operatori d'andamento Incr* $\nearrow$, *Decr* $\searrow$ e *Const* $\rightleftharpoons$, associati, rispettivamente, alla crescenza, alla decrescenza e alla costanza dell'andamento nel tempo dei valori d'appartenenza ad un concetto.

Estendo, in 3.2.2, il modello alle logiche crisp con le qualified number restrictions $\mathcal{Q}$ e passo, in 3.2.3, alle logiche fuzzy. Una rivalutazione degli operatori *SubsNext* e *SubsPrev* porta alla ridefinizione in ambito fuzzy degli *operatori d'andamento Incr* $\nearrow$, *Decr* $\searrow$ e *Const* $\rightleftharpoons$, come gli operatori associati anche alla monotonia dell'andamento nel tempo, per andamenti mediamente crescenti, decrescenti o costanti, rispettivamente.

In 3.2.4, estendo il modello alle logiche fuzzy con $\mathcal{Q}$ ed infine, in 3.2.5, estendo $\mathcal{Q}$ in $\mathcal{Q}^{+}$, cioè ammetto anche i quantificatori fuzzy, che consentono le espressioni quantificate $QR.C$ e $QC \sqsubseteq D$. Definisco i nuovi *quantificatori temporali* fuzzy Q_T e la restrizione Q_{now}; in seguito analizzo come trattare le query dipendenti dal tempo e le query che coinvolgano una sussunzione.

Alla fine di ogni paragrafo del capitolo 3 sono proposti e, di volta in volta, aggiornati quattro esempi di applicazione del modello, in quattro differenti ambiti: il commerciale, il bancario, il controllo dei sistemi, ed il monitoraggio ambientale.

Nel capitolo 4 espongo la progettazione e l'implementazione di un modulo di reasoning automatico che permetta la valutazione di query contenenti i costrutti più originali definiti nel capitolo 3. Esso consente la creazione e la gestione di una *Knowledge Base*, l'esecuzione di alcuni tipi di query e la persistenza della *Knowledge Base* in formato XML.

Il capitolo 5 contiene la progettazione e l'implementazione di alcune applicazioni dimostrative, in cui si usa il modulo di reasoning oggetto del capitolo 4.

In 5.1 descrivo un modulo di editing che permette la creazione e configurazione delle *MBox* e *TBox* di una *Knowledge Base* e la sua esportazione in formato xml. In 5.2 si presentano tre casi di monitoring, successivamente dettagliati.

In 5.3 descrivo un modulo di monitoring in tempo reale, ove si ha una continuativa e periodica importazione dell'*ABox* dal sistema dinamico in esecuzione. Nello specifico, si riprende e si

implementa un esempio già analizzato, quello di un sistema dinamico di tipo preda – predatori. Motivi di efficienza impongono l'esecuzione degli algoritmi iterativi incrementali, ove si valutino operatori che ammettano un tal tipo di semantica.

In 5.4 descrivo un modulo di monitoring che importa l'*ABox* da un database relazionale con una frequenza così bassa da permettere l'esposizione grafica dei dati importati nell'*ABox*, mediante diagramma a punti, in modo che l'utente possa avere anche una percezione visiva degli andamenti. Utilizzo questo tipo di monitoring in 5.4.1, per la valutazione di query connesse al problema del riscaldamento del pianeta e in 5.4.2 per la valutazione di query a carattere sanitario.

Nel capitolo 6 do alcuni cenni conclusivi ed alcune idee su possibili sviluppi futuri.

Nel capitolo 7 ci sono alcune appendici. In 7.1 enuncio e dimostro un teorema che non ho inserito nella parte teorica per non appesantirla, ma che consente maggior efficienza computazionale in implementazione. In 7.2 do un cenno schematico sul percorso di un reasoner nell'esecuzione di una query di sussunzione generale. In 7.3 si trova qualche brano più significativo di codice d'implementazione e documentazione.

All'inizio di ogni capitolo è presente un'introduzione più dettagliata.

2

Lo stato dell'arte

In questo capitolo prendo in considerazione lo stato dell'arte riguardo agli argomenti coinvolti nella mia ricerca.

Nel paragrafo 2.1 do una panoramica delle logiche DL fuzzy: nate come estensione delle logiche crisp, di cui aumentano l'espressività dei concetti e ruoli, richiedono una sintassi modificata e una revisione delle operazioni di base sugli insiemi. In 2.1.1, diverse norme sono state definite in letteratura, ricercando la compatibilità con differenti proprietà crisp, poiché non tutte possono essere mantenute: io adotto quelle tra le più comunemente usate.

In 2.1.2, descrivo sintassi e semantica dei linguaggi logici f- $\mathcal{ALC}$ e da essi arrivo alla sintassi e semantica delle logiche $f\_\mathcal{SHOIQ}$(D). In 2.1.2.1 do alcuni cenni sui diversi aspetti dell'operatore di sussunzione, di come la sua interpretazione può fornire il grado d'implicazione tra concetti oppure il grado di inclusione tra insiemi. Spesso le due interpretazioni coincidono.

In 2.1.3 do una definizione di una *Knowledge Base*, in generale per logiche fuzzy, dei suoi componenti e dei problemi d'inferenza di base.

Nel paragrafo 2.2 introduco il concetto dei quantificatori fuzzy, assoluti e relativi. L'approccio basato sulla cardinalità porta al problema della cardinalità degli insiemi fuzzy e alle proposte dei diversi autori, in particolare alla cardinalità scalare e alla cardinalità fuzzy. Mi soffermo sulla non convessità della cardinalità fuzzy e sulla definizione di misura di cardinalità fuzzy assoluta ED di un insieme e poi relativa ER di un insieme rispetto ad un altro, proposte da Delgado et al (5). Da essi, passo alla valutazione GD delle asserzioni quantificate, come valutazione del grado di compatibilità tra il quantificatore e la cardinalità fuzzy, assoluta o relativa, del corrispondente insieme o della corrispondente coppia d'insiemi. Il metodo GD sarà da me adottato per le valutazioni delle espressioni quantificate.

Nel paragrafo 2.3, descrivo brevemente le logiche DL temporali. In 2.3.1 è presente una veloce rassegna delle caratteristiche delle diverse DL temporali crisp. Mi soffermo poi sulla proposta $\mathcal{ALCQIT}$ di Artale e Franconi(6) e sulle relative sintassi e semantica degli operatori temporali. In 2.3.2 descrivo i contributi in letteratura alle DL temporali fuzzy.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

2.1 Le logiche descrittive fuzzy

Nelle logiche classiche, *crisp*, l'asserzione che un individuo appartenga o non ad un dato concetto è netta. L'uso del simbolo ∈ oppure ∉ non lascia alternative. Per ogni dato concetto C, si determina una dicotomia del mondo, tra individui che vi appartengono e individui che non vi appartengono. Nelle logiche *fuzzy*, esiste invece una flessibilità nell'appartenenza ai concetti, e può esserci una funzione d'appartenenza o *membership-function* μ, a valori in [0, 1], che la descrive. Ogni individuo x può appartenere ad un concetto anche parzialmente, con un grado α contenuto in [0, 1]. I due estremi $\alpha = 0$ e $\alpha = 1$ corrispondono rispettivamente agli ∈ e ∉ crisp. Quindi, passiamo da una descrizione (crisp) del mondo in cui, per ogni individuo x e per ogni concetto C esiste l'alternativa dicotomica $x \in C$ (x appartiene a C) oppure $x \notin C$ (x non appartiene a C), alla descrizione fuzzy in cui $\langle x: C, \alpha \rangle$ (x appartiene a C con un grado α).

Gli insiemi fuzzy sono stati introdotti da Zadeh (7), come un modo per trattare concetti vaghi come pressione_bassa, velocità_alta, ecc. Un *insieme fuzzy* C, rispetto ad un universo U, può essere caratterizzato da una funzione di appartenenza, *membership function* μ_A: $U \rightarrow [0,1]$, che assegna un grado d'appartenenza $\mu_C(x)$ ad ogni elemento x di U. $\mu_C(x)$ dà una stima dell'appartenenza di x nell'insieme C. In linguaggio logico, diremo che $\mu_C(x)$ esprime il *grado di verità* dell'asserzione 'x *è* C'. Gli insiemi crisp sono un caso particolare degli insiemi fuzzy, dove la *membership function* può assumere i due soli valori in {0,1}.

Rigorosamente, potremmo parlare solo di sottoinsiemi fuzzy definiti su un universo U, poiché l'universo U è crisp e non fuzzy (consideriamo gli individui appartenenti all'universo sempre in modo totale o non appartenenti affatto); gli individui stessi sono individui crisp (pur partecipando in modo fuzzy ai vari concetti). Inoltre, ci fermiamo alla descrizione fuzzy di primo livello, perché potremmo rendere fuzzy anche le membership functions, ecc. Infine, l'universo che prenderò in considerazione sarà, salvo quando esplicitato diversamente, sempre finito.

Le logiche DL fuzzy trattano i concetti come insiemi fuzzy e i ruoli come relazioni binarie fuzzy: di conseguenza, ogni individuo appartenente ad un concetto od ogni coppia di individui partecipanti ad un ruolo, lo sono con un certo grado. Dovremo scrivere allora, non più $a \in C$ o $(a,b) \in R$, come nelle logiche crisp, ma $\langle a:C, n \rangle$ e $\langle (a,b):R, n \rangle$, rispettivamente; n è il *grado di verità*, compreso in [0,1], dato eventualmente dal valore assunto dalla membership function: $n = \mu_C(a)$ oppure $n = \mu_R(a,b)$. Per semplicità, scriverò spesso $C(a)$ intendendo $\mu_C(a)$ oppure $R(a,b)$ intendendo $\mu_R(a,b)$.

2.1.1 Connettivi logici. Norme

L'estensione fuzzy di concetti e operatori crisp non è sempre immediata e univoca. Le operazioni di base sugli insiemi fuzzy, la negazione, l'unione e l'intersezione, possono essere definite in modi diversi, a seconda delle proprietà che si vuole che abbiano, in aggiunta alle condizioni fondamentali. Purtroppo, non tutte le proprietà valide nelle logiche crisp sono conservate passando alle logiche fuzzy.

Il complemento di un concetto A è chiamato *negation* ed è un nuovo concetto, dato da una membership function $(\neg A)(x) = n(A(x))$, con $n : [0,1] \rightarrow [0,1]$. L'unione di due concetti A e B è chiamata *triangular co-norm* o *s-norm,* ed è il nuovo concetto la cui membership function è data

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

dalla funzione s : [0,1] × [0,1] → [0,1] con $(A \vee B)(x) = s(A(x), B(x))$. L'intersezione di due concetti A e B è detta *triangular – norm* o *t-norm*, concetto dato dalla funzione t: [0,1] × [0,1] → [0,1] con $(A \wedge B)(x) = t(A(x), B(x))$. Importante è anche l'operazione di *implicazione* fuzzy, che è una funzione i: [0,1] × [0,1] → [0,1] che assegna un valore di verità alla formula $A \rightarrow B$.

In questa mia ricerca userò, salvo quando indicato diversamente, le norme di Zadeh, dette anche norme standard, per la negazione, la t-norm e la s-norm, che sono le maggiormente utilizzate. Per l'implicazione, invece, preferirò usare la i-norm di Gödel.

Le norme che userò sono definite dalla Tabella 2-1.

Le norme di Zadeh soddisfano la legge di de Morgan $a \wedge b = \neg(\neg a \vee \neg b)$: $\forall a,b \in [0,1]$, $s(a, b) = n(t(n(a), n(b)))$. Inoltre le operazioni ¬, ∧ e ∨ sono idempotenti.

L'implicazione di Gödel appartiene alla famiglia delle implicazioni residuali $i_R(x, y) = sup\{z \in [0,1] \mid t(x, z) \leq y\}$. Esse derivano da una generalizzazione della proprietà dell'implicazione crisp: $a \rightarrow b = \neg a \vee b = max\{c \in \{0,1\} \mid a \wedge c \leq b\}$. Per queste i-norme, se $a \leq b$ allora $i_R(a, b) = 1$, indipendentemente dalla t-norma usata.

Purtroppo, altre proprietà valide nel caso crisp, non sono soddisfatte da questa scelta di operatori logici fuzzy. Ad esempio, $\forall a,b \in [0,1]$, $i(a, b) = n(t(a, n(b)))$ non vale e dunque $\forall R.C$ non è più equivalente a $\neg\exists R.\neg C$.

Una data Description Logic è definita dall'insieme delle formule valide e dalla loro semantica.

2.1.2 Sintassi e Semantica

Le logiche $\mathcal{ALC}$ (Attributive Language with Complement) costituiscono il linguaggio descrittivo minimo che includa la negazione e la disgiunzione.

Le regole sintattiche per le $\mathcal{ALC}$ fuzzy, *f*-$\mathcal{ALC}$, sono in Tabella 2-2.

Date le $\mathcal{ALC}$, se si aggiungono i ruoli transitivi $\mathcal{R}_+$, si ottengono i linguaggi $\mathcal{S} = \mathcal{ALCR}_+$; se ad essi si aggiungono l'inclusione tra ruoli $\mathcal{H}$, i nominali $\mathcal{O}$, i ruoli inversi $\mathcal{I}$ e le restrizioni in numero $\mathcal{N}$, si ottengono i linguaggi $\mathcal{SHOIN}$; aggiungendo le restrizioni in numero qualificate $\mathcal{Q}$, si ottengono

Tabella 2-1 norme usate

norma	sintassi	semantica	
negation	¬ x	1 - x	Zadeh
t-norm	x ∧ y	*min*(x, y)	Zadeh
s-norm	x ∨ y	*max*(x, y)	Zadeh
i-norm	x → y	if x ≤ y then 1 else y	Gödel

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}(\mathrm{D})$

Tabella 2-2 regole sintattiche per le f-$\mathcal{ALC}$

Concetti:	$C, D \rightarrow \top \mid \bot \mid A \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \forall R.C \mid \exists R.C$
Ruoli:	R
Asserzioni:	$\alpha \rightarrow \langle \mathrm{a} : C, \mathrm{r} \rangle \mid \langle (\mathrm{a},\mathrm{b}) : R, \mathrm{r} \rangle$
Assiomi:	$\tau \rightarrow \langle C \sqsubseteq D, \mathrm{r} \rangle$

gli $\mathcal{SHOIQ}$; si può dotare il tutto dei domini concreti (D) e renderli fuzzy, $f\_$, per raggiungere gli $f\_\mathcal{SHOIQ}(\mathrm{D})$. Nella Tabella 2-3 è presente uno schema della sintassi e relativa semantica.

La semantica è assegnata dall'*interpretazione* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, dove $\Delta^{\mathcal{I}}$ *o* $U^{\mathcal{I}}$, è il *dominio o universo* dell'interpretazione e $\cdot^{\mathcal{I}}$ è la funzione d'interpretazione, che mappa ogni concetto in un insieme fuzzy d'individui, ogni individuo, istanza di un concetto, in un grado d'appartenenza compreso in [0,1] e ogni coppia d'individui, istanza di un ruolo, anch'essa in un grado d'appartenenza compreso in [0,1].

Nella semantica, si mantengono separate le classi di 'oggetti' dai domini concreti. I *domini concreti* sono, nelle DL crisp, tipi di dati concreti, quali insiemi numerici oppure string. Nelle DL fuzzy, un *dominio fuzzy concreto* è una coppia $D = \langle \Delta_D , \phi_D \rangle$, dove Δ_D è un dominio d'interpretazione concreto e ϕ_D è l'insieme dei *predicati fuzzy* d *di dominio concreto*, con una arità predefinita n=1, 2 ed un'interpretazione fissata d^{D}: $\Delta_D^{\,n} \rightarrow [0,1]$.

I *concetti concreti* sono concetti definiti, come i predicati fuzzy di dominio concreto, su un dominio concreto e hanno una membership function esplicita. Esempi possono essere il concetto *Veloce*, sul dominio concreto delle velocità, oppure *Costoso*, sul dominio concreto dei prezzi, oppure *Giovane*, sul dominio concreto delle età, se si definisce esplicitamente la loro membership function.

La notazione "(D)" nella notazione $f\_\mathcal{SHOIQ}(\mathrm{D})$ si riferisce appunto ai domini concreti.

Una nota sulla semantica della *composizione* di ruoli. Se abbiamo, in generale, due ruoli $R_1 : X \times Z \rightarrow [0,1]$ *e* $R_2 : Z \times Y \rightarrow [0,1]$, la loro composizione $R_1 . R_2 : X \times Y \rightarrow [0,1]$, è definita come $R_1 . R_2^{\mathcal{I}}(\mathrm{x}, \mathrm{y}) = sup_{\mathrm{z} \in \mathbb{Z}} R_1^{\mathcal{I}}(\mathrm{x}, \mathrm{z}) \wedge R_2^{\mathcal{I}}(\mathrm{z}, \mathrm{y})$, che è un'estensione della composizione tra ruoli crisp $R_1 : X \times Z \rightarrow \{0,1\}$ *e* $R_2 : Z \times Y \rightarrow \{0,1\}$, dove la loro composizione $R_1 . R_2 : X \times Y \rightarrow \{0,1\}$, è definita come $R_1 . R_2^{\mathcal{I}}(\mathrm{x}, \mathrm{y})$ iff $\exists \mathrm{z} \mid R_1^{\mathcal{I}}(\mathrm{x}, \mathrm{z}) \wedge R_2^{\mathcal{I}}(\mathrm{z}, \mathrm{y})$.

2.1.2.1 Inclusione e sussunzione

verso $f\_\,\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Tabella 2-3 sintassi e semantica per i linguaggi $f\_\,\mathcal{SHOIQ}$(D)

	Sintassi	*Semantica*
Concetti: C, D →	$\top$ |	$\top(\mathrm{x})^{\mathcal{I}}=1$
	$\bot$ |	$\bot(\mathrm{x})^{\mathcal{I}}=0$
concetto atomico	A |	$A^{\mathcal{I}}(\mathrm{x})\in[0,1]$
intersezione	$C\sqcap D$ |	$(C\sqcap D)^{\mathcal{I}}(\mathrm{x})=C^{\mathcal{I}}(\mathrm{x})\wedge D^{\mathcal{I}}(\mathrm{x})$
unione	$C\sqcup D$ |	$(C\sqcup D)^{\mathcal{I}}(\mathrm{x})=C^{\mathcal{I}}(\mathrm{x})\vee D^{\mathcal{I}}(\mathrm{x})$
negazione	$\neg C$ |	$(\neg C)^{\mathcal{I}}(\mathrm{x})=\neg C^{\mathcal{I}}(\mathrm{x})$
quantific. universale	$\forall R.C$ |	$(\forall R.C)^{\mathcal{I}}(\mathrm{x})=inf_{y\in\Delta^{\mathcal{I}}}R^{\mathcal{I}}(\mathrm{x},y)\rightarrow C^{\mathcal{I}}(y)$
quantific. esistenziale	$\exists R.C$ |	$(\exists R.C)^{\mathcal{I}}(\mathrm{x})=sup_{y\in\Delta^{\mathcal{I}}}R^{\mathcal{I}}(\mathrm{x},y)\wedge C^{\mathcal{I}}(y)$
qualified number restriction	$\geq \mathrm{n}\,R.C$ |	$(\geq \mathrm{n}R.C)^{\mathcal{I}}(\mathrm{x})=sup_{y_1,\ldots,y_n\in\Delta^{\mathcal{I}}}\wedge_{i=1}^{n}R^{\mathcal{I}}(\mathrm{x},y_i)\wedge C^{\mathcal{I}}(y_i)$
	$\leq \mathrm{n}\,R.C$ |	$(\leq \mathrm{n}R.C)^{\mathcal{I}}(\mathrm{x})=\neg(\geq \mathrm{n}+1R.C)^{\mathcal{I}}(\mathrm{x})$
nominali	$\{a_1,\ldots,a_{\mathrm{n}}\}$ |	$\{a_1,\ldots,a_{\mathrm{n}}\}^{\mathcal{I}}(\mathrm{x})=\vee_{i=1}^{n}a_i^{\mathcal{I}}=\mathrm{x}$
concetti concreti	F	$F^{\mathcal{I}}(\mathrm{v})=\mu_F(\mathrm{v})$
Ruoli: R→	T |	$T^{\mathcal{I}}(y,\mathrm{x})\in[0,1]$
inversa	T^{-} |	$(T^{-})^{\mathcal{I}}(\mathrm{x},y)=T^{\mathcal{I}}(y,\mathrm{x})$
composizione	$R_1.R_2$	$R_1.R_2^{\mathcal{I}}(\mathrm{x},\mathrm{y})=sup_{\Delta^{\mathcal{I}}}R_1^{\mathcal{I}}(\mathrm{x},\mathrm{z})\wedge R_2^{\mathcal{I}}(\mathrm{z},\mathrm{y})$
Asserzioni: α →	$\langle \mathrm{a}:C,\mathrm{r}\rangle$ |	$C^{\mathcal{I}}(\mathrm{a}^{\mathcal{I}})\geq \mathrm{r}$
	$\langle(\mathrm{a},\mathrm{b}):R,\mathrm{r}\rangle$	$R^{\mathcal{I}}(\mathrm{a}^{\mathcal{I}},\mathrm{b}^{\mathcal{I}})\geq \mathrm{r}$
Assiomi: τ →	$\langle C\sqsubseteq D,\mathrm{r}\rangle$ |	$\mathrm{r}=inf_{x\in\Delta^{\mathcal{I}}}(C^{\mathcal{I}}(\mathrm{x})\rightarrow D^{\mathcal{I}}(\mathrm{x}))$
	$\langle R\sqsubseteq T,\mathrm{r}\rangle$ |	$\mathrm{r}=inf_{x,y\in\Delta^{\mathcal{I}}}(R^{\mathcal{I}}(\mathrm{x},\mathrm{y})\rightarrow T^{\mathcal{I}}(\mathrm{x},\mathrm{y}))$
functional	*fun(R)* |	$\forall\mathrm{x}\forall\mathrm{y}\forall\mathrm{z}\,(R(\mathrm{x},\mathrm{y})\wedge R(\mathrm{x},\mathrm{z})\rightarrow \mathrm{y}=\mathrm{z})$
transitive	*trans(R)*	$\forall\mathrm{x}\forall\mathrm{y}\forall\mathrm{z}\,(R(\mathrm{x},\mathrm{y})\wedge R(\mathrm{y},\mathrm{z})\rightarrow R(\mathrm{x},\mathrm{z}))$
(per ogni x, $y\in\Delta^{\mathcal{I}}$, $\mathrm{v}\in\Delta_{\mathrm{D}}$)		

Assiomi di inclusione: la semantica dell'assioma d'inclusione $\langle C\sqsubseteq D,\mathrm{r}\rangle^{\mathcal{I}}$, $\mathrm{r}=inf_{x\in\Delta^{\mathcal{I}}}(C^{\mathcal{I}}(\mathrm{x})\rightarrow D^{\mathcal{I}}(\mathrm{x}))$, corrisponde all'espressione $\forall\mathrm{x}\in\Delta^{\mathcal{I}}\,(C^{\mathcal{I}}(\mathrm{x})\rightarrow D^{\mathcal{I}}(\mathrm{x}))\geq \mathrm{r}$. Nel caso in cui valga che $\forall\mathrm{x}C^{\mathcal{I}}(\mathrm{x})\leq D^{\mathcal{I}}(\mathrm{x})$, si ha che, con le i-norme residuali, $r=1$ e ciò corrisponde all'inclusione tra insiemi più intuitiva. Questa definizione, introdotta da Straccia (8), è una traduzione diretta della formula FOL $\forall\mathrm{x}.F_C(\mathrm{x})\rightarrow F_D(\mathrm{x})$ ed ha il vantaggio di essere una formula fuzzy, nel senso che assegna un grado di verità, compreso in [0,1], all'inclusione tra due concetti fuzzy.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Conviene spendere due parole in più riguardo al *grado di sussunzione* tra due concetti.

$$\langle A \sqsubseteq B,\ s\rangle,\ s \in [0,1] \qquad \text{2-1}$$

Per mantenere l'accordo con la definizione crisp di sussunzione, imponiamo la condizione che

$$\text{if } A(\mathrm{x}) \leq B(\mathrm{x})\ \forall \mathrm{x} \in U \text{ then } S(A,B) = 1. \qquad \text{2-2}$$

La sussunzione può essere vista

1. come una relazione di implicazione fuzzy, oppure
2. come una relazione di sottoinsieme tra insiemi fuzzy.

2.1.2.1.1 Grado d'implicazione. *Implication degree*.

Dati A e B insiemi fuzzy definiti su un universo finito U, quando si considera la sussunzione (2-1), come un'implicazione, allora il grado d'implicazione è (omettendo l'apice di interpretazione):

$$s_1 = \inf_{\mathrm{x}\in U}(A(\mathrm{x}) \rightarrow B(\mathrm{x})) = \inf_{\mathrm{x}\in U} i(A(\mathrm{x}), B(\mathrm{x})) \qquad \text{2-3}$$

Dove $i(a,b)$ è una generica funzione di implicazione fuzzy (i-norma).

Come abbiamo visto, tra le varie i-norme esistenti in letteratura, quelle che soddisfano la condizione (2-2) appartengono al gruppo delle *i-norme residuali* $i_R(a,b) = sup\ \{\mathrm{x} \in [0,1]: t(a,\mathrm{x}) \leq b\}$, che derivano da una generalizzazione della proprietà dell'implicazione crisp: a $\rightarrow$ b = ¬a ∨ b = $max\{\mathrm{c} \in \{0,1\}: \mathrm{a} \wedge \mathrm{c} \leq \mathrm{b}\}$.

Per queste i-norme, se $a \leq b$ allora $i_R(a,b) = 1$, indipendentemente dalla t-norma usata.

Esempi di $\mathrm{i_R}$ sono la $\mathrm{i_L}$ di Lukasiewicz $i_L(a,b) = min\ (1, 1-a+b)$ e la $\mathrm{i_G}$ di Gödel $i_G(a,b) = sup\ \{\mathrm{x} \in [0,1]: min(a,\mathrm{x}) \leq b\}$.

Utilizzando quest'ultima, definiamo il *grado di sussunzione* $\mathrm{s_1}$, $\langle A \sqsubseteq B \geq s_1\rangle$, tra due concetti fuzzy A, B, considerando la sussunzione una relazione d'implicazione fuzzy:

$$s_1 = inf_{\mathrm{x}\in U}\ sup\ \{c \in [0,1]: min(A(\mathrm{x}), \mathrm{c}) \leq B(\mathrm{x})\}. \qquad \text{2-4}$$

2.1.2.1.2 Grado di sottoinsieme. *Subsethood degree*.

Inclusione di insiemi fuzzy. Dati A e B insiemi fuzzy definiti su un universo finito U, si dice che A è sottoinsieme di B, $A \subseteq B$, se e solo se

$$A(\mathrm{x}) \leq B(\mathrm{x})\ \forall \mathrm{x} \in U \qquad \text{2-5}$$

Sotto le operazioni standard sugli insiemi fuzzy, $A \subseteq B$ se e solo se $A \cap B = A$ e $A \cup B = B$.

Eseguiamo una valutazione euristica: se adottiamo provvisoriamente la cardinalità scalare σ_count (9) come cardinalità di un insieme fuzzy A: $|A| = \sigma\_count(A) = \sum_{\mathrm{x}\in U} A(\mathrm{x})$, allora possiamo definire il grado di sottoinsieme, *subsethood degree* $S(A, B)$, per ogni coppia di insiemi fuzzy definiti su un universo U finito, come

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$$S(A,B) = \frac{1}{|A|}(|A| - \sum_{x \in U} max[0, A(x) - B(x)]), \quad 0 \leq S(A,B) \leq 1 \qquad 2\text{-}6$$

che corrisponde ad 1- il grado di violazione della condizione $\forall x(A(x)) \leq B(x)$.

Otteniamo:

$$S(A,B) = \frac{1}{|A|}(\sum_{x \in U} \min[A(x), B(x)]) = \frac{|t(A,B)|}{|A|} = \frac{|A \cap B|}{|A|} \qquad 2\text{-}7$$

dove $t(a,b) = min\ (a,b)$ è la funzione d'intersezione fuzzy standard o di Zadeh.

Più in generale, possiamo esprimere il *grado di sottoinsieme* $S(A, B)$*, subsethood degree,* come:

$$S(A,B) = \frac{|A \cap B|}{|A|} = \frac{|t(A,B)|}{|A|} \qquad 2\text{-}8$$

Dove $|\cdot|$ è una generica funzione di cardinalità scalare di un insieme fuzzy e $t(\cdot,\cdot)$ è una generica funzione d'intersezione fuzzy (t-norma).

Esistono in letteratura t-norme diverse da quella di Zadeh, che però, non soddisfano la condizione imposta. Es: la t-norma di Lukasiewicz $t_L(a,b) = max(a+b-1, 0)$ non dà a quando $a \leq b$. Questo è un altro motivo per utilizzare la t-norma standard, di Zadeh.

Si può definire il *grado di sussunzione* s_2 tra due *concetti* fuzzy A, B, considerandoli particolari insiemi fuzzy, scrivendo $\langle A \sqsubseteq B, s_2 \rangle$, come il loro grado di sottoinsieme:

$$s_1 = S(A,B) = \frac{|A \cap B|}{|A|} = \frac{|t(A,B)|}{|A|} \qquad 2\text{-}9$$

Utilizzando, invece della cardinalità scalare, una cardinalità fuzzy relativa, ad esempio il metodo ER (5) (Definizione 2), il grado di sussunzione si può generalizzare ulteriormente in

$$s_2 = ER(B,A). \qquad 2\text{-}10$$

2.1.3 Knowledge Base

Si chiama *Terminological Box* o *TBox* $\mathcal{T}$ l'insieme degli assiomi terminologici τ, $\mathcal{T}=\{\tau\}$;

si chiama *Assertion Box* o *ABox* $\mathcal{A}$ l'insieme delle asserzioni α, $\mathcal{A}=\{\alpha\}$;

si chiama *Meta Box* o *Meta Ontologia* o *MBox* $\mathcal{M}$ l'insieme di definizioni che riguardano il linguaggio, le meta-definizioni delle strutture che si useranno per definire le specifiche ontologie.

Una Base di Conoscenza o *Knowledge Base* od *Ontologia* $\mathcal{K}$ è la coppia costituita da una *TBox* e una *ABox*, $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

Ogni *TBox*, propriamente, contiene anche una *Upper Ontology* od Ontologia Superiore, la quale definisce quei concetti così generali da poter esser considerati comuni a tutte le ontologie.

2.1.3.1 Problemi d'inferenza di base

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Su una *Knowledge Base* $\mathcal{K}$ sono possibili ragionamenti inferenziali per ottenere nuova conoscenza. I problemi d'inferenza di base sono:

consistency: controllare se $\mathcal{K}$ è consistente, cioè soddisfacibile, cioè non presenta contraddizioni;

subsumption: $\mathcal{K} \vDash C \sqsubseteq D$? : dare una struttura tassonomica alla conoscenza;

equivalence: $\mathcal{K} \vDash C = D$? : controllare se due concetti sono equivalenti;

graded instantiation: $\mathcal{K} \vDash \langle a : C, r \rangle$? : controllare se un individuo è istanza di un concetto con grado almeno r;

best true value bound: $|\, a : C \,|_{\mathcal{K}} = \sup\{ r \mid \mathcal{K} \vDash \langle a : C, r \rangle \}$: trovare il grado massimo per cui un individuo è istanza di un concetto;

top-n retrieval: $\text{top}(n)\{\langle a, r\rangle \mid r = |\, a : C \,|_{\mathcal{K}} \}$: estrarre l'insieme delle n istanze di un concetto *C*, col massimo grado d'appartenenza.

2.2 Le logiche descrittive fuzzy e quantificate $\mathcal{ALCQ}^{+}$

Zadeh (7) ha mostrato che i quantificatori imprecisi presenti in ogni linguaggio parlato possono essere definiti nelle description logics usando insiemi fuzzy. Incorporando questi nel linguaggio logico e fornendo le espressioni per definire la loro semantica, si può realizzare un sistema di rappresentazione della conoscenza potente, con potere espressivo più vicino al modo di pensare umano.

Le asserzioni quantificate fuzzy sono asserzioni imprecise che concernono il numero o la percentuale di oggetti che verificano una certa proprietà. Si possono classificare in due tipi. Due esempi sono:

tipo I: 'circa una ventina di studenti è alta';

tipo II: 'quasi tutti gli studenti sono alti'.

Quantificatori come '*circa una ventina*', '*non più di una trentina', 'più o meno sette o più',* '*qualcuno*', che compaiono nelle asserzioni del tipo I e che indicano il numero, sono detti *quantificatori assoluti*; quantificatori come '*quasi tutti', 'circa la metà', 'la maggior parte', 'una piccola parte',* che compaiono nelle asserzioni del tipo II e che indicano una frazione o percentuale, sono detti *quantificatori relativi*.

I quantificatori assoluti sono numeri fuzzy, cioè insiemi fuzzy, definiti solitamente sugli interi non-negativi $\mathbb{Z}$ oppure sui reali non-negativi $\mathbb{R}_0^{+}$, nel caso di insieme continuo di individui; io mi limiterò a considerare universi discreti, dunque quantificatori assoluti su $\mathbb{Z}$. I quantificatori relativi sono invece percentuali fuzzy, cioè insiemi fuzzy sull'intervallo unitario $[0,1] \subseteq \mathbb{R}$.

L'approccio basato sulla *cardinalità*, per la valutazione della verità di un'asserzione quantificata, calcola il grado di compatibilità tra il quantificatore fuzzy che esprime la quantificazione e la cardinalità dell'insieme fuzzy che rappresenta l'asserzione.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Alcuni autori, quali De Luca et al. (9), Ralescu (10), Dubois et al.(11), Wygralak (12), hanno proposto una *cardinalità scalare* (es., $\sigma\_$ *count* di De Luca et al. (9)) per gli insiemi fuzzy, cioè un numero intero o reale, come estensione della cardinalità crisp, che è sempre un numero intero.

Altri autori, quali Zadeh(13)(14) e successivamente gli stessi Dubois et al.(11) e Wygralak (15), ritengono più adeguato rappresentare la cardinalità di un insieme fuzzy con una *cardinalità fuzzy*, cioè un altro insieme fuzzy, contenente più valori possibili di cardinalità, interi non-negativi, associati ad un grado di verità o di verosimiglianza.

Delgado et al. (16) hanno espresso argomenti per ritenere che questa cardinalità fuzzy, di un insieme fuzzy F, non debba necessariamente essere convessa. La convessità per una cardinalità imporrebbe che se F ha un certo valore di cardinalità $n1$ con grado di verità $v1$ e un altro valore di cardinalità $n2$ con grado di verità $v2$, ogni altro valore di cardinalità intermedio dovrebbe avere grado di verità intermedio. Come valore di cardinalità s'intende comunque un numero intero non-negativo, il numero di individui appartenenti al concetto.

Un esempio semplice ma significativo è dato da un universo U, composto dai tre individui mary, lucy e john, e il concetto *Blonde* (tipicamente fuzzy) degli individui 'biondi'. Se *ABox* = {⟨mary: *Blonde*, 1.0⟩, ⟨lucy: *Blonde*, 0.5⟩ e ⟨john: *Blonde*, 0.5⟩} (Figura 2-1), cioè se mary è 'completamente' bionda, mentre gli altri due sono solo 'mezzi' biondi, quanti individui biondi ci sono in U ? Ovvero, qual è la cardinalità dell'insieme *Blonde*? La risposta non è univoca ed è rappresentata da un insieme fuzzy contenente i valori possibili da 0 a 3 (il numero d'individui in U, che è crisp). Essendo mary *Blonde* con valore massimo, almeno lei dovrà essere presa in considerazione, quindi il valore di verosimiglianza per la cardinalità 1 sarà maggiore di 0. Come tener conto degli altri due? Se consideriamo anche lucy *Blonde*, allora necessariamente dovremo considerare *Blonde* anche john, che ha lo stesso grado di *Blonde* di lucy. Quindi o li consideriamo entrambi o nessuno dei due. Insomma la cardinalità di *Blonde* dovrebbe essere o 1 oppure 3. Per essere più concreti, supponendo di dover comprare dei cappelli per i 'biondi', il numero dei cappelli da comprare dipenderà dalla soglia che si impone per considerare un individuo 'biondo'. Certamente compreremo almeno un cappello per mary che è *Blonde* al massimo grado, ma se abbassiamo la soglia fino a comprendere anche lo 0.5, cioè chi è *Blonde* a metà, allora ne compreremo altri due, per un totale di tre. Quindi 1 o 3, senza valore intermedio.

Nell'insieme fuzzy della cardinalità di *Blonde*, allora n=2 avrà grado di verità 0, mentre n=1 e n=3 avranno grado di verità > 0. Dunque l'insieme non è convesso.

Delgado et al. (16) propongono un metodo di valutazione della cardinalità (*cardinalità assoluta*) di un insieme fuzzy F, che si basa sulla distribuzione di probabilità di F decomposto nei suoi α – *cuts* e che ha il vantaggio di non essere alternativo ma di assomigliare molto ad altri metodi proposti (es., Zadeh, Dubois), nel senso che essi appartengono ad una stessa famiglia di cardinalità fuzzy.

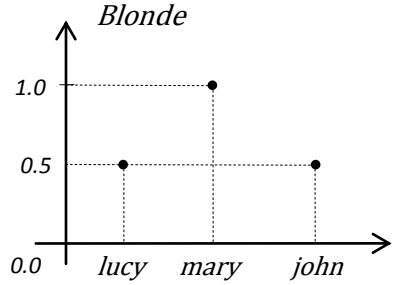

Figura 2-1 il concetto fuzzy *Blonde*

Definizione 1. La misura di *cardinalità fuzzy assoluta* *ED* (16) di un insieme fuzzy F è definita, per ogni $0 \leq k \leq |supp(F)|$, come

$$ED(F, k) = \begin{cases} \alpha_i - \alpha_{i+1}, & per\ k = |F_{\alpha_i}|\ e\ \alpha_i \in \Lambda(F) \\ 0, & altrimenti \end{cases} \qquad 2\text{-}11$$

dove $supp(F) = \{x \in U \mid F(x) \neq 0\}$ è il supporto di F, $F_{\alpha i}$ è l'$\alpha - cut$ di F al livello α_i e $\Lambda(F) = \{\alpha_1, .., \alpha_p\} \cup \{1\}$ è l'insieme dei livelli di F, ordinati con l'ordinamento $\alpha_i \geq \alpha_{i+1}$ per ogni $i \in \{1, .., p\}$ e $\alpha_{p+1} = 0$ (Figura 2-2).
Riguardo alla valutazione della *cardinalità relativa* di una coppia di insiemi F e G, la cardinalità relativa di G rispetto ad F, cioè la percentuale degli elementi di F che sono in G, nel caso di insiemi crisp, essa è data dal rapporto

$$RC(G/F) = \frac{|F \cap G|}{|F|} \qquad 2\text{-}12$$

Nel caso di insiemi fuzzy, Delgado (7) propone il seguente metodo *ER*, che estende *ED*.

Definizione 2. La *cardinalità fuzzy relativa* *ER* (16) di un insieme G rispetto ad un insieme F è l'insieme $ER(G/F)$, definito per ogni $q \in [0,1] \cap \mathbb{Q}$ come

$$ER(G/F, q) = \sum_{\alpha_i} | C(G/F, \alpha_i) = q^{(\alpha_i - \alpha_{i+1})}\ , \qquad 2\text{-}13$$

dove $C(G/F, \alpha)$ è definito come il valore razionale $C(G/F, \alpha) = RC(G_\alpha/F_\alpha) = |(F \cap G)_\alpha|/|F_\alpha|$, $\alpha_i \in \Lambda(F) \cup \Lambda(F \cap G) = \{\alpha_1, .., \alpha_p\}$ *e* $\alpha_i > \alpha_{i+1}$ per ogni $i \in \{1, .., p\}$ e $\alpha_0 = 1$, $\alpha_{p+1} = 0$. Nel caso di F crisp e a cardinalità $|F| = n$, si ha che $ER(G/F, k/n) = ED(G, k)$, che dimostra che *ER* estende *ED*.

La valutazione di un'asserzione quantificata richiede di valutare il grado di compatibilità tra il quantificatore e la cardinalità fuzzy, assoluta o relativa, del corrispondente insieme o della corrispondente coppia d'insiemi.

L'asserzione quantificata è sempre del tipo *"Q degli F sono G"*, dove F può eventualmente essere anche l'intero universo *U*. Q può essere del tipo *assoluto* Q^A, oppure *relativo* Q^R. Nel caso di Q^A, la compatibilità dovrà essere valutata con la cardinalità fuzzy assoluta dell'intersezione $F \cap G$, $ED(F \cap G)$, che risponde alla domanda *"quanti degli F sono in G? "*; nel caso di Q^R, la compatibilità dovrà essere valutata con la cardinalità fuzzy relativa di G rispetto ad F, $ER(G/F)$, che risponde alla domanda *"quale percentuale degli F sono in G? "*. Si ottiene il seguente metodo *GD:*

Definizione 3. (4) Dati due insiemi F, G e un quantificatore Q, il metodo *GD* ottiene la valutazione dell'asserzione quantificata *"Q degli F sono G"*:

$$GD_{Q^A}(G/F) = \sum_{\alpha_i \in \Lambda(G/F)} (\alpha_i - \alpha_{i+1}) Q^A(|(F \cap G)_{\alpha_i}|)\ ,\ \text{per } Q = Q^A \text{ assoluto,} \qquad 2\text{-}14$$

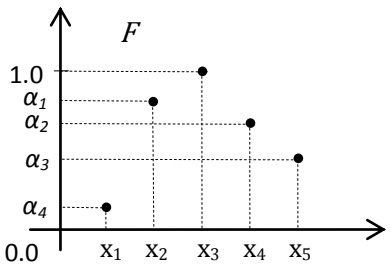

Figura 2-2 un insieme fuzzy F e i suoi α - cuts

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$$GD_{Q^R}(G/F) = \sum_{\alpha_i \in \Lambda(G/F)} (\alpha_i - \alpha_{i+1}) Q^R \left(\frac{\left|(F \cap G)_{\alpha_i}\right|}{\left|F_{\alpha_i}\right|} \right) \text{, per } Q{=}Q^R \text{ relativo.} \qquad \text{2-15}$$

N.B. nel caso di quantificatori relativi, GD non è definito se F non è normalizzato, poiché esiste almeno un valore di α ($\alpha = 1$), per cui $|F_\alpha| = 0$. Si può scegliere allora tra due metodi: o normalizzare F e scalare con lo stesso fattore anche $F \cap G$, oppure, per ogni particolare quantificatore, assegnare nella sua definizione il valore che esso assume sulla singolarità 0/0. Delgado (14) ritiene questa seconda via più conveniente, perché più economica, nel caso di valutazioni ripetute su insiemi in generale non normalizzati.

Un caso particolare importante è la valutazione dell'asserzione quantificata *"tutti gli F sono G"*: $ER(G / F, 1)$. In questo caso si sommano i contributi per cui $C(G/F, \alpha) = |(F \cap G)_\alpha| / |F_\alpha| = 1$, ovvero per cui $F_{\alpha i} \subseteq G_{\alpha i}$ e si ottiene:

$$GD(G/F) = \sum_{\alpha_i \in \Lambda(G/F) | (F)_{\alpha_i} \subseteq (G)_{\alpha_i}} (\alpha_i - \alpha_{i+1}). \qquad \text{2-16}$$

2.3 Le logiche descrittive temporali

La logica temporale lineare $\mathcal{LTL}$, v. Emerson (17), è usata con successo per descrivere, specificare proprietà e verificare i sistemi concorrenti. Le formule di tale logica sono interpretate su modelli che sono sequenze infinite di stati e si definiscono modalità temporali per asserire proprietà di queste sequenze.

2.3.1 Logiche descrittive temporali crisp

Sono stati proposti, nell'ambito delle Description Logics, diversi approcci per la rappresentazione e il ragionamento su concetti dipendenti dal tempo, v. Artale (1). Essi differiscono:

- sull'ontologia del tempo: possono adottare un punto di vista basato su una nozione del tempo come *punto* oppure come *intervallo*. Questo secondo punto di vista è adottato dalla logica modale temporale a intervalli, combinata con alcune restrizioni $\mathcal{HS}$, v. Halpern (18). Il mio approccio è invece basato su una nozione puntuale del tempo.
- sul modo in cui è aggiunta la *dimensione temporale*: si può adottare una nozione *esplicita* del tempo e gli operatori usati nella costruzione di nuove formule ne dipendono esplicitamente oppure la nozione del tempo può rimanere *implicita* e il linguaggio rappresenta una sorta di sequenze di cambiamenti di stato, come sequenza di eventi, v. Devanbu (19).
- Quando è adottata una nozione esplicita del tempo, il punto di vista può ancora essere suddiviso in esterno ed interno, v. Finger (20).
    - Nel punto di vista *esterno*, gli individui sono unici ma mutevoli, nel senso che possono assumere stati diversi nei diversi istanti di tempo. L'interpretazione è dipendente dal tempo e coglie dall'esterno, istante per istante, in quale stato si trovino gli individui.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

- Nel punto di vista *interno*, invece, ogni individuo è visto come una collezione di componenti, costituenti lo stato dell'individuo in ogni istante particolare, v. Haarslev (21). Questa è la mia impostazione, che illustrerò diffusamente.

Artale e Franconi (6) hanno introdotto una estensione temporale $\mathcal{ALCQIT}$ di $\mathcal{ALCQI}$, con un punto di vista esplicito ed esterno, con una struttura temporale discreta, lineare e illimitata. Ulteriore estensione, molto più espressiva è stata poi studiata per le $\mathcal{DLR}$ (22). Restando in $\mathcal{ALCQIT}$, gli operatori introdotti per descrivere gli aspetti dipendenti dal tempo dei concetti sono presi dalla $\mathcal{LTL}$, Linear Temporal Logics: essi sono *Until* $\mathcal{U}$, *Since* $\mathcal{S}$ e i loro derivati sometime in the past (*somepast*) $\Diamond^{-}$, sometime in the future (*somefuture*) $\Diamond^{+}$, always in the past (*allpast*) $\Box^{-}$, always in the future (*allfuture*) $\Box^{+}$. Queste le regole sintattiche aggiunte:

$$C, D \rightarrow C\,\mathcal{U}\,D \mid C\,\mathcal{S}\,D \mid \Diamond^{-}C \mid \Diamond^{+}C \mid \Box^{-}C \mid \Box^{+}C \qquad \text{2-17}$$

Gli operatori $\Diamond^{-}$, $\Diamond^{+}$, $\Box^{-}$, $\Box^{+}$ sono operatori derivati. Infatti sono definiti $\Diamond^{+}C \equiv \top\,\mathcal{U}C$; $\Diamond^{-}C \equiv \top\mathcal{S}C$ e dualmente per $\Box^{+}$ e $\Box^{-}$.

Si assume una struttura temporale $\mathcal{T} = (\mathcal{P}, <)$, data da un insieme $\mathcal{P}$ di *punti di tempo* e da un ordinamento stretto $<$ su $\mathcal{P}$. Un'*interpretazione temporale* è definita sopra $\mathcal{T}$ come una coppia $\mathcal{M} \equiv \langle \mathcal{T}, \mathcal{I} \rangle$, dove $\mathcal{I}$ è una funzione che associa ad ogni t $\in \mathcal{P}$ un'interpretazione non temporale standard $\mathcal{I}(t) \equiv \langle \Delta^{\mathcal{I}}, R_0^{\mathcal{I}(t)}, .., C_0^{\mathcal{I}(t)}, .. \rangle$, che soddisfi le regole d'interpretazione standard $\mathcal{ALCQI}$ ed in più le seguenti:

$$(C\mathcal{U}D)^{\mathcal{I}(t)} = \{i \in \Delta^{\mathcal{I}} \mid \exists v.\, v > t \wedge D^{\mathcal{I}(v)}(i) \wedge \forall w.(t < w < v) \rightarrow C^{\mathcal{I}(w)}(i)\} \qquad \text{2-18}$$

$$(C\mathcal{S}D)^{\mathcal{I}(t)} = \{i \in \Delta^{\mathcal{I}} \mid \exists v.\, v < t \wedge D^{\mathcal{I}(v)}(i) \wedge \forall w.(v < w < t) \rightarrow C^{\mathcal{I}(w)}(i)\} \qquad \text{2-19}$$

In parole semplici, la semantica dell'operatore *Until* dice che $C\,\mathcal{U}D$ è (al tempo t) il concetto insieme degli individui per i quali esiste nel loro futuro (rispetto a t) un istante v in cui essi sono D e in tutto il periodo compreso tra t e v essi sono C.

Analogamente, la semantica dell'operatore *Since* dice che $C\,\mathcal{S}D$ è (al tempo t) il concetto insieme degli individui per i quali esiste nel loro passato (rispetto a t) un istante v in cui essi sono D e in tutto il periodo compreso tra t e v essi sono C. Conseguentemente, la semantica degli operatori *sometime* $\Diamond^{\pm}$ dice che $\Diamond^{\pm}\,C$ è l'insieme degli individui che al tempo t hanno nel loro passato (rispettivamente, nel loro futuro) almeno un istante in cui sono C, mentre, per *always*, $\Box^{\pm}C$ è l'insieme degli individui che, al tempo t, sono stati (o, rispettivamente, saranno) sempre C.

Da notare che siamo in ambito crisp, quindi l'appartenenza degli individui ai concetti è netta: o sì o no, o 0 o 1. *Until* e *Since* sono definiti in modo forte, cioè escludendo gli estremi negli intervalli temporali. Sarebbe possibile anche una definizione debole, comprensiva degli estremi, che permetterebbe ugualmente la derivazione degli altri operatori introdotti, purché deboli anch'essi,

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

mentre non permetterebbe la derivazione di altri operatori tipici delle logiche temporali lineari $\mathcal{LTL}$ come *next* e *last*.

Questo *Until* è, più propriamente, secondo le $\mathcal{LTL}$, *Until*-in-the-future e *Since* è *Since*-in-the-past. Si potrebbero definire anche gli operatori *Until*-in-the-past e *Since*-in-the-future. La struttura temporale illimitata lascia delle difficoltà di decisione: ad esempio per escludere che un individuo appartenga ad un *Until*, essendo il futuro di t in $\mathcal{P}$ illimitato.

2.3.2 Logiche descrittive temporali fuzzy

Si è studiato l'aspetto temporale per le DL, anche in ambito fuzzy. I principali contributi sono rivolti al monitoraggio e al controllo automatico dei sistemi.

Ben Lamine (23) introduce una logica fuzzy temporale basata sugli stati, piuttosto che sugli eventi. Si tratta quindi di una logica dove la dimensione temporale resta implicita. Le proposizioni del linguaggio si riferiscono agli stati; la valutazione e la validazione adottano un metodo incrementale, che rende più efficiente la loro esecuzione al volo, on the fly.

Sono ammessi concetti fuzzy e asserzioni fuzzy. Trattandosi di un sistema di monitoraggio deputato alla decisione e al controllo, la valutazione deve poi essere defuzzificata. Si osserva una sequenza finita di stati, di lunghezza fissata a priori secondo considerazioni empiriche, e si valutano le proposizioni logiche sull'intera sequenza.

Le formule sono formate coi connettivi booleani *and* e *not* e gli operatori temporali *next*, *always*, *sometime*, *until*.

Le formule sono interpretate su modelli della forma $\langle w,\pi \rangle$, dove w è una sequenza infinita di stati del mondo $\{w_0,w_1,..\}$ e π è una funzione a valori reali che valuta le proposizioni negli stati w_i. Data una proposizione p ed uno stato w_i, $\pi(p, w_i)$ restituisce il valore di verità di p nello stato del mondo w_i. in generale, dunque, il valore di verità di una proposizione dipende dallo stato.

Per uno stato w_i in un modello $M = \langle w,\pi \rangle$, per la proposizione p e per le formule f_1 e f_2 si hanno le regole d'interpretazione date dalla Tabella 2-4.

Nella Tabella 2-4, *s* e *t* sono le s-norm e t-norm standard (rispettivamente *max* e *min*). Si nota come il valore di verità restituito dalla funzione d'interpretazione π dipende non solo dallo stato attuale w_i, ma anche da una sequenza di stati terminante in w_i. La lunghezza della sequenza è, come detto, fissata a priori a seguito di considerazioni empiriche. Più in generale, si può fare una media pesata (*OWA*, ordered weighted average) dei valori lungo la sequenza. Le definizioni permettono l'utilizzo di un algoritmo di valutazione progressivo.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Tabella 2-4 regole d'interpretazione delle formule per il modello temporale di Ben Lamine

$\pi(\neg p, w_i) = 1 - \pi(p, w_i)$	*negazione*
$\pi(f_1 \wedge f_2, w_i) = t(\pi(f_1, w_i), \pi(f_2, w_i))$	*t-norm*
$\pi(f_1 \vee f_2, w_i) = s(\pi(f_1, w_i), \pi(f_2, w_i))$	*s-norm*
$\pi(\oplus f, w_i) = \pi(f, w_{i+1})$	*next*
$\pi(\Box f, w_i) = t(\pi(f, w_i), \pi(\Box f, w_{i+1}))$	*always*
$\pi(\Diamond f, w_i) = s(\pi(f, w_i), \pi(\Diamond f, w_{i+1}))$	*sometime*
$\pi(f_1 U f_2, w_i) = s(\pi(f_2, w_i), \pi(f_1 U f_2, w_{i+1}))$	*until*

3

Modelli matematici

In questo capitolo, introduco i modelli matematici che costituiscono l'oggetto centrale di questa tesi.

Nel paragrafo 3.1 descrivo il modello $\mathcal{ALC}$ fuzzy, con quantificatori fuzzy $\mathcal{Q}^{+}$, denominato $f-\mathcal{ALCQ}^{+}(\mathrm{D})$. I quantificatori sono considerati insiemi fuzzy e le espressioni quantificate sono interpretate con il metodo GD (4).

Introduco poi una funzione parametrizzata lineare a tratti *Trap*, di forma genericamente trapezoidale, che sarà utilizzata nella definizione delle membership function dei concetti concreti e dei quantificatori. Espressioni quantificate valide saranno dei due tipi $QR.C$ e $Q\,C \sqsubseteq D$. Accenno anche all'opportunità di definire una quasi – sussunzione o soft-subsumption $\tilde{\sqsubseteq}$.

Il paragrafo 3.2 costituisce il cuore di questo lavoro, dove definisco il mio modello temporale. Il modello temporale è definito progressivamente, in un percorso di accrescimento dell'espressività. Inizio in 3.2.1, con le logiche crisp; estendo in 3.2.2 alle logiche crisp con le qualified number restrictions $\mathcal{Q}$; passo, in 3.2.3, alle logiche fuzzy e, in 3.2.4, alle logiche fuzzy con $\mathcal{Q}$ ed infine, in 3.2.5, al modello combinato temporale su logiche fuzzy e con i quantificatori fuzzy $\mathcal{Q}^{+}$.

In 3.2.1 definisco una nuova *Upper Ontology* introducendo una partizione dell'Universo – inteso come l'insieme degli individui – in *Static* e *Dynamic* e di *Dynamic* in *ULine* ed *Event*, dove *ULine* ed *Event* sono disgiunti ma strettamente interdipendenti; inoltre, è introdotto un dominio concreto numerico discreto, il quale induce una struttura, a semantica temporale, che determina un ordinamento sugli individui. Accanto agli ordinari concetti statici, posso definire così sintassi e semantica di nuovi *concetti dinamici* (temporali) ottenuti con gli operatori temporali $\Box$, $\Diamond$, $\mathcal{U}$ e $\mathcal{S}$. Si pongono alcune questioni inedite, quali quelle sull'ereditarietà o trasmissibilità dei concetti e ruoli tra *ULine* ed *Event*.

Considerando più strettamente l'insieme ordinato degli *Event* corrispondenti ad una stessa *ULine*, si arriva naturalmente alla definizione degli operatori *Next* $\oplus$ e *Prev* $\ominus$ e, successivamente, *SubsNext* e *SubsPrev*, ottenuti applicando la sussunzione. L'osservazione che esiste una relazione tra la crescenza nel tempo dei valori d'appartenenza a un concetto e

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

SubsNext e tra la decrescenza nel tempo dei valori e *SubsPrev* porta alla definizione dei nuovi *operatori d'andamento Incr* ↗, *Decr* ↘ e *Const* ⇌. Sono infine proposti quattro esempi di applicazione del modello, in quattro differenti ambiti: il commerciale, il bancario, il controllo dei sistemi dinamici, ed il monitoraggio ambientale.

In 3.2.2 arricchisco il modello precedente ammettendo espressioni con le restrizioni in numero qualificate $\mathcal{Q}$ e mostro subito l'applicazione su uno degli esempi già proposti.

In 3.2.3 estendo il modello alle logiche fuzzy, senza operare grandi variazioni formali, ma ottenendo un notevole arricchimento semantico. Una rivalutazione degli operatori *SubsNext* e *SubsPrev* porta alla ridefinizione, in ambito fuzzy, degli *operatori d'andamento Incr* ↗, *Decr* ↘ e *Const* ⇌, come gli operatori associati anche alla monotonia dell'andamento, per andamenti mediamente crescenti, decrescenti o costanti, rispettivamente. L'estensione al fuzzy è poi trasmessa ai modellini d'esempio.

In 3.2.4 estendo con $\mathcal{Q}$, come fatto nel crisp, e ne osservo i vantaggi.

Nel modello combinato, in 3.2.5, estendo $\mathcal{Q}$ in $\mathcal{Q}^{+}$, cioè ammetto anche i quantificatori fuzzy, che consentono le espressioni quantificate $QR.C$ e $QC \sqsubseteq D$. Definisco i nuovi *quantificatori temporali* fuzzy Q_T e vedo che con essi acquistano semantica anche espressioni come $Q_T C$, che quindi ammetto nella sintassi. Definisco Q_{now} come restrizione di Q_T ed in seguito dimostro due teoremi, a supporto della coerenza del modello. Analizzo poi come trattare le query dipendenti dal tempo e considero la semantica delle query che coinvolgano una sussunzione. Infine arricchisco i modellini d'esempio con le nuove conquiste di linguaggio.

3.1 Modello quantificato fuzzy $f\_\mathcal{ALCQ}^{+}$(D)

Come abbiamo visto, sono stati introdotti, (5)(4), quantificatori come insiemi fuzzy, per descrivere e valutare espressioni qualificate, del tipo I: 'Q *degli* U *sono* G' e del tipo II: 'Q *degli* F *sono* G'.

Questo metodo permette di essere applicato nelle logiche descrittive per estendere le quantificazioni esistenziali ∃, assolute ∀ e le restrizioni in numero ≥n, ≤n fino ad ammettere espressioni come $QR.C$, dove Q rappresenta un qualunque quantificatore relativo od assoluto. $QR.C$ è dunque il concetto contenente quegli individui per cui Q delle loro relazioni R vanno in C. Ad esempio, un'espressione lecita è $(Most)buys.CheapProduct$ che rappresenta il concetto di coloro la cui maggior parte degli acquisti sono prodotti economici. Per ogni individuo x, si può valutare il suo grado d'appartenenza al concetto $QR.C$, considerando R_x, la proiezione di R su x, come un insieme fuzzy. R_x è definito come $\langle y: R_x \rangle = \langle (x,y) : R \rangle$ per ogni y in U.

L'espressione $QR_x.C$ si interpreta come 'Q *degli elementi di* $R_x^{\mathcal{I}}$ *sono* $C^{\mathcal{I}}$', nel caso di $Q = Q^R$ relativo, e 'Q *degli elementi di* $U^{\mathcal{I}}$ *sono* $R_x^{\mathcal{I}}$ *e* $C^{\mathcal{I}}$', nel caso di $Q = Q^A$ assoluto, e si valuta col metodo GD (Definizione 3):

$$(Q^A R.C)^{\mathcal{I}}(x) = GD_{Q^A}\left(R^{\mathcal{I}}{}_x \cap C^{\mathcal{I}} / U^{\mathcal{I}}\right), \text{ dove } Q^A \text{ è quantificatore assoluto;} \qquad 3\text{-}1$$

$$(Q^R R.C)^{\mathcal{I}}(x) = GD_{Q^R}\left(C^{\mathcal{I}} / R^{\mathcal{I}}{}_x\right), \text{ dove } Q^R \text{ è quantificatore relativo.} \qquad 3\text{-}2$$

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Si può introdurre, per comodità, una semplice membership-function standard parametrizzata, lineare a tratti, grazie alla quale definire ogni concetto fuzzy concreto, quantificatori compresi. Essa assume la forma di un trapezio che può anche essere degenere.

Definizione 4. Si denomina *Trapezium* o *Trap* la funzione parametrica
Trap (*tipo*, a , b, c, d, *min*, *max*): $\mathbb{R} \to [0,1]$,

con *tipo* $\in$ {*abs*, *rel* }; a, b, c, d, *min*, *max* $\in \mathbb{R}$; *min* $\leq$ a $\leq$ b $\leq$ c $\leq$ d $\leq$ *max* ; così definita:

$$Trap\,(tipo, a, b, c, d, min, max)(x) = \begin{cases} 0 & \text{per } min \leq x < a \\ \frac{x-a}{b-a} & \text{per } a \leq x < b \\ 1 & \text{per } b \leq x \leq c \\ \frac{d-x}{d-c} & \text{per } c < x \leq d \\ 0 & \text{per } d < x \leq max \end{cases} . \qquad 3\text{-}3$$

min e *max* sono gli estremi del dominio concreto numerico. Nel caso che *Trap* sia un quantificatore, *rel* indica che è relativo, *abs* che è assoluto. Nel caso di concetto concreto, si pone *tipo* = *abs*. Poiché per quantificatori relativi il valor minimo è sempre 0 e il valor massimo è sempre 1, quando si scrive *Trap* (a, b, c, d), (Figura 3-2 a), si intende

$$Trap\,(a, b, c, d) = Trap\,(rel, a, b, c, d, 0, 1), \qquad 3\text{-}4$$

mentre, quando si scrive Trap (a, b, c, d, *min*, *max*), (Figura 3-2 b), con 6 parametri, si intende

$$Trap\,(a, b, c, d, min, max) = Trap\,(abs, a, b, c, d, min, max) \qquad 3\text{-}5$$

Ad esempio, si possono descrivere insiemi crisp, come *Trap* (a, a, d, d, *min*, *max*), insiemi triangolari come *Trap* (a, b, b, d, *min*, *max*), spalla sinistra (left shoulder) come *Trap* (*min, min*, c, d, *min, max*), spalla destra (right shoulder) come *Trap* (a, b, *max*, *max*, *min*, *max*) (Figura 3-3).

Si possono definire concetti come *Young* = *Trap* (0, 0, 16, 25, 0, 120), su un dominio di età, o come *Cold* = *Trap* (-10, -5, 5, 10, -40, 60), su un dominio di temperature (Figura 3-4),

e quantificatori come *Most* = *Trap* (0.4, 0.6, 1, 1), *Between*(~2)*and*(~4) = *Trap* (1, 2, 4, 5, 0, *max*), *Exactly*(n) = *Trap* (n, n, n, n, *min*, *max*) , *All* = ⟨*Exactly*(1), *rel* ⟩ ≡ *Trap* (1, 1, 1, 1) (Figura 3-5).

Così sono descrivibili i quantificatori delle logiche crisp:

$$\forall \equiv All \equiv Trap\,(rel, 1,1,1,1), \qquad 3\text{-}6$$

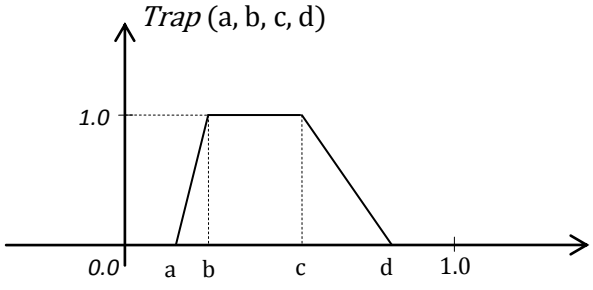

Figura 3-2 a *Trap* relativo

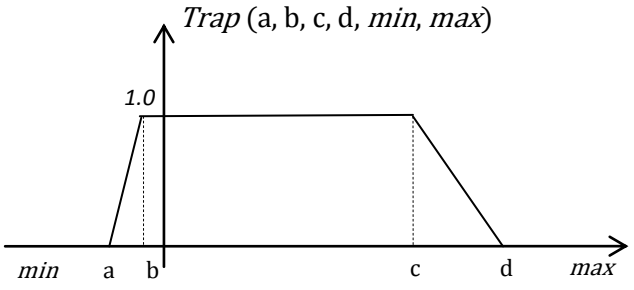

Figura 3-2 b *Trap* assoluto

$\exists \equiv Trap\,(1, 1, max, max, 0, max)$ assoluto, oppure 3-7

$\exists \equiv \neg Exactly(0) \equiv \neg\, Trap\,(0, 0, 0, 0)$ relativo, 3-8

$\geq n \equiv MoreThan(n) \equiv Trap\,(n, n, max, max, 0, max)$, 3-9

$\leq n \equiv LessThan(n) \equiv Trap\,(0, 0, n, n, 0, max)$. 3-10

Anche l'espressione di sussunzione $A \sqsubseteq B$ si può pensare come caso particolare di un'espressione più generale $(Q)A \sqsubseteq B$ di *sussunzione quantificata*.

La sussunzione $A \sqsubseteq B$ corrisponde all'affermazione che tutti gli elementi di A sono anche elementi di B o, meglio, $A(x) \leq B(x)$ per ogni x del dominio. Se si tratta di una query, si vuol sapere con quale grado questa relazione sia vera.

Adottiamo il metodo GD (Definizione 3) (2-16) per valutare la sussunzione $A \sqsubseteq B$:

$$(A \sqsubseteq B)^{\mathcal{I}} = GD(B/A). \quad 3\text{-}11$$

Se utilizziamo il quantificatore All definito sopra, allora è immediato vedere l'equivalenza $A \sqsubseteq B \equiv (All)A \sqsubseteq B$, quando per $((Q)A \sqsubseteq B)^{\mathcal{I}}$ si usi il metodo $GD_Q(B/A)$ (2-14, 2-15).

$((Q^A)A \sqsubseteq B)^{\mathcal{I}} = GD_{Q^A}(B^{\mathcal{I}}/A^{\mathcal{I}})$, dove Q^A è quantificatore assoluto; 3-12

$((Q^R)A \sqsubseteq B)^{\mathcal{I}} = GD_{Q^R}(B^{\mathcal{I}}/A^{\mathcal{I}})$, dove Q^R è quantificatore relativo. 3-13

Ora, notiamo che la sussunzione $A \sqsubseteq B$ è fuzzy nel senso che ammette un valore di verità

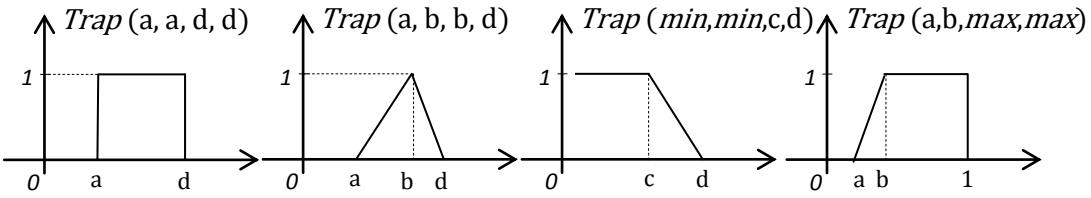

Figura 3-3 insieme crisp, insieme triangolare, left shoulder, right shoulder

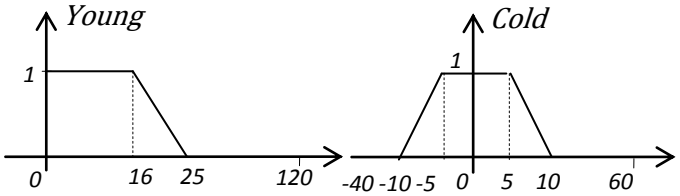

Figura 3-4 *Young, Cold*

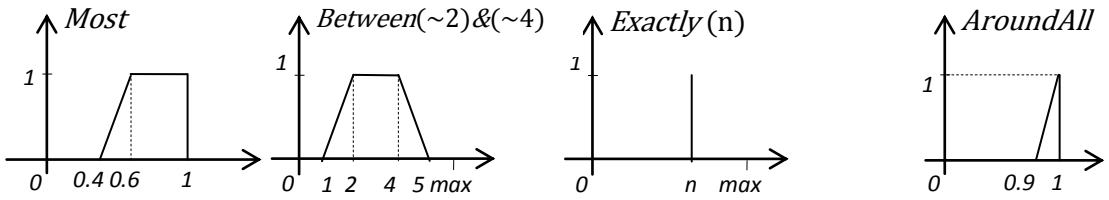

Figura 3-5 *Most*, *Between*(~2)&(~4), *Exactly*(n)

Figura 3-6 *AroundAll*

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

appartenente all'intero intervallo unitario; d'altronde, è una sussunzione 'perfetta', poiché il suo valore di verità rappresenta la quota d'inclusione perfetta tra i due insiemi, ogni 'quasi' inclusione dando contributo nullo (Figura 3-7).

Si potrebbe allora definire una 'quasi sussunzione':

Definizione 5. Si definisce *quasi – sussunzione* o *soft – subsumption* l'espressione

$$A \tilde{\sqsubseteq} B \equiv (AroundAll)A \sqsubseteq B \qquad 3\text{-}14$$

con $AroundAll = Trap(0.9, 1, 1, 1)$ (Figura 3-6).

La quasi sussunzione $A \tilde{\sqsubseteq} B$ è più fuzzy, nel senso che tiene conto anche della 'quasi inclusione' tra insiemi. Ha la proprietà che comunque vale 1 solo quando anche $A \sqsubseteq B$ vale 1. Il terzo esempio della Figura 3-7 avrebbe un valore $(A \tilde{\sqsubseteq} B) > 0$.

Rimane, ovviamente, l'arbitrarietà nella scelta del quantificatore nella definizione della quasi sussunzione, ad es. $AroundAll$ = 1 iff All =1 o è preferibile allargare $AroundAll$?

D'altronde, questo è un problema di tutti gli insiemi fuzzy e in particolare per tutti i quantificatori. Gli insiemi fuzzy sono stati introdotti per descrivere con una gradualità i concetti imprecisi. Purtroppo anche questa gradualità è un compromesso, che si esplicita in una membership function: bisogna convenire sui valori, sulla forma e sulla pendenza di questa gradualità. Ma non mi dilungherò su questa questione.

Si noti che la definizione delle membership function dei concetti concreti e dei quantificatori appartiene, propriamente, a quella che abbiamo chiamato *Meta Ontologia*.

3.2 Modello temporale

3.2.1 Modello temporale crisp $\mathcal{SHOINT}$ (D)

Desidero introdurre un nuovo approccio per rappresentare gli individui che abbiano una propria estensione temporale. Chiamerò questi individui *dinamici*, mentre gli altri *statici.* Gli individui dinamici sono gli individui le cui proprietà o i cui ruoli possono modificarsi al variare di un parametro temporale. Ad esempio, l'individuo 'maria' o 'il mio cane' sono presumibilmente individui dinamici, mentre 'idrogeno' è individuo statico.

Definisco una *UpperOntology* od Ontologia Superiore, od *O*ntologia con la *O* maiuscola, da distinguersi dalle *o*ntologie particolari, che descrivono aspetti o porzioni particolari del mondo, le

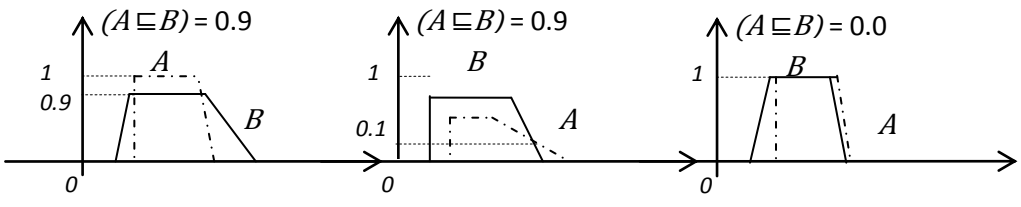

Figura 3-7 esempi d'inclusione 'perfetta'

*o*ntologie con la *o* minuscola. Tutte le *o*ntologie dovranno far riferimento alla *O*ntologia nelle loro descrizioni e ragionamenti.

Definisco i due concetti (crisp) *Static* e *Dynamic*, a intersezione nulla, che partizionano l'Universo. Ogni individuo o è statico oppure dinamico.

$$\top \sqsubseteq Static \sqcup Dynamic \qquad 3\text{-}15$$

$$Static \sqcap Dynamic \sqsubseteq \bot \qquad 3\text{-}16$$

Definisco il concetto (crisp) di LineaUniverso, *ULine*.

$$\top \sqsubseteq ULine \qquad 3\text{-}17$$

Definisco il concetto (crisp) di Evento, *Event*.

$$\top \sqsubseteq Event \qquad 3\text{-}18$$

Dynamic è partizionato in *ULine* ed *Event*: ogni individuo *Dynamic* è *ULine* oppure *Event* (Figura 3-8):

$$Dynamic \equiv ULine \sqcup Event \qquad 3\text{-}19$$

$$ULine \sqcap Event \sqsubseteq \bot \qquad 3\text{-}20$$

Ad ogni Linea Universo associo un sottoinsieme (crisp) di *Event*, tramite un ruolo (crisp) *event*, di cui esista l'inversa, che chiamo *uline*. Ogni Linea Universo deve avere almeno un evento (Figura 3-9).

$$event: ULine \rightarrow Event \qquad 3\text{-}21$$

[1]

$$uline: Event \rightarrow ULine \qquad 3\text{-}22$$

$$uline \equiv event^{-} \qquad 3\text{-}23$$

$$\top \sqsubseteq \forall event.Event \qquad 3\text{-}24$$

$$\top \sqsubseteq \forall uline.ULine \qquad 3\text{-}25$$

$$ULine \sqsubseteq \exists event.\top \qquad 3\text{-}26$$

Ogni evento deve avere una e una sola corrispondente linea Universo.

$$Event \sqsubseteq \exists uline.\top \qquad 3\text{-}27$$

$$fun(uline) \qquad 3\text{-}28$$

Gli elementi di *ULine* inducono dunque su *Event* una partizione, in corrispondenza biunivoca con essi.

[1] Convenzione 1: da qui in avanti, adotto la convenzione per cui, dato un ruolo *r*, con l'espressione "*r*: $A \rightarrow B$" indico che A è il dominio di r e B è il suo codominio (range). In altre parole, "*r*: $A \rightarrow B$" equivale alle due espressioni DL: $\top \sqsubseteq \forall r.B$, $A \sqsubseteq \exists r.\top$, che a volte ripeto per chiarezza.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$$\exists \mathcal{F} = \{ X_i \} \mid Event = \cup_i X_i \wedge X_i \sqcap X_j = \phi \wedge i \neq j \wedge X_i, X_j \in \mathcal{F} \wedge$$

$$\forall X_i \in \mathcal{F} \ \exists \tilde{x}_i \in ULine \mid X_i = event(\tilde{x}_i) \wedge \forall \tilde{x}_i \in ULine \exists \ X_i \in \mathcal{F} \ \mid X_i = event(\tilde{x}_i). \qquad 3\text{-}29$$

Gli eventi *Event* corrispondono agli eventi spaziotemporali della fisica, ma potrebbero anche essere chiamati *realizzazioni*, *incarnazioni*, *manifestazioni, istanziazioni o avatar* della Linea Universo di un individuo. La Linea Universo *ULine*, composta di Eventi, a sua volta si potrebbe vedere anche come, in se stessa, *l'astrazione* o il *se'* dell'individuo. Un Evento non può appartenere contemporaneamente a due Linee Universo. Questo rimarrà vero anche dopo la successiva 'fuzzificazione' del sistema: non si potrà mai appartenere un po' ad una linea universo e un po' ad un'altra, ovvero due linee universo (l'insieme degli *Event* corrispondenti) non si intersecheranno mai. In altre parole, rimango aderente ad una visione 'classica' della fisica, che pur ammette la relatività e la fisica quantistica. Nella fisica quantistica, la creazione e l'annichilazione di particelle corrispondono a creazione e terminazione istantanee di linee universo. In ogni istante una particella appartiene ad una e una sola linea universo. Altre ipotesi più 'ardite' potranno essere prese in considerazione in un eventuale ricerca futura. Mi limiterò a considerare la dimensione temporale e non quella spaziale e quindi i miei eventi sono eventi puramente temporali.

Definisco il predicato concreto *time*, che associa ad ogni individuo *Static* o *Event* uno e un solo

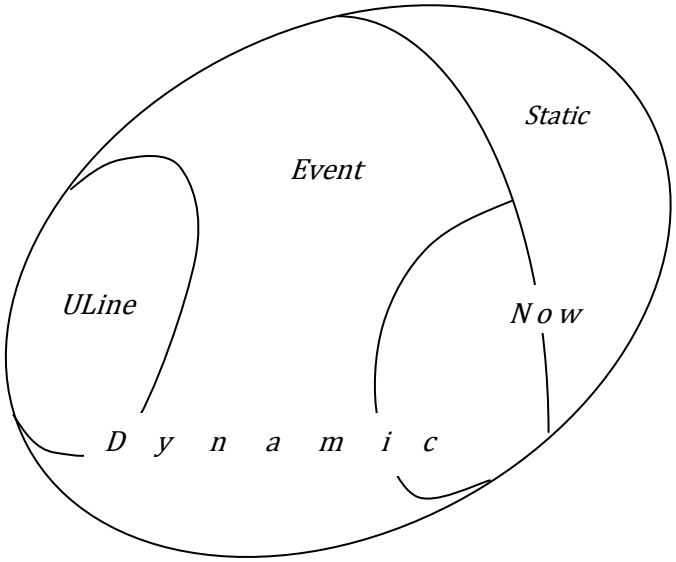

Figura 3-8 la partizione concettuale dell'Universo

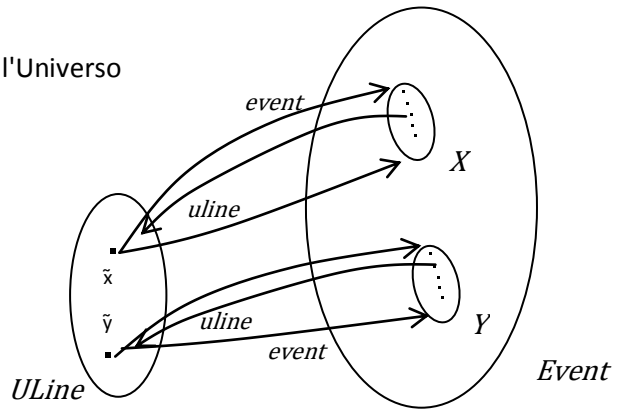

Figura 3-9 Linee Universo ed Eventi

numero reale, che assumerà la semantica dell'istante di tempo di esistenza dell'individuo.

$$time : \top \rightarrow Real^{2} \qquad 3\text{-}30$$

$$fun(time) \qquad 3\text{-}31$$

$$Static \sqcup Event \sqsubseteq \exists time.\top \qquad 3\text{-}32$$

Resta definito il sottoinsieme *Time* dei reali assegnati da *time*.

$$Time \equiv Real \sqcap \exists time^{-}.\top \qquad 3\text{-}33$$

L'assunzione per questa tesi è che esista un meccanismo, nel sistema descrittivo, per cui, ad intervalli regolari di tempo, oppure negli istanti più significativi, siano creati eventi *Event* nuovi e ad essi sia assegnato, oltre alle proprietà proprie, anche l'istante di tempo, tramite il predicato concreto *time*. In tal modo, l'insieme *Time* degli istanti assegnati è un insieme discreto di punti. Esso, inoltre, può sempre essere supposto limitato, sia inferiormente, ad esempio dall'età dell'universo, $t_0 \gtrsim -14 \times 10^9$ anni, che superiormente, da una soglia che si può sempre porre nei calcoli sul futuro, per sistemi che abbiano significato informatico o fisico. *Time* è quindi discreto e finito.

L'ordinamento esistente in *Real* induce in modo naturale un ordinamento '$<$' in *Time* ; con esso possiamo definire la struttura temporale $\mathcal{T} = \langle Time, < \rangle$.

L'ordinamento su *Time* induce a sua volta un ordinamento sugli individui statici od eventi, a cui diamo il significato di precedenza temporale e contemporaneità.

Definizione 6. Dati due individui $x, y \in Static \sqcup Event$, diremo che

'x *precede* y' e indicheremo '$x < y$' se e solo se $time(x) < time(y)$;

'x è contemporaneo a y*'* e indicheremo '$x =_t y$ ' se e solo se $time(x) = time(y)$;

'x *precede o è contemporaneo a* y ' e indicheremo '$x \leq y$' se e solo se $time(x) \leq time(y)$;

'x *segue* y *'* e indicheremo '$x > y$ ' se e solo se 'y *precede* x *'*.

Dato questo ordinamento, ci saranno utili più avanti i seguenti ruoli.

Definizione 7. Si definiscono i ruoli *concurrent*, *precedent* e *subsequent*, tutti $Static \sqcup Event \rightarrow Static \sqcup Event$, con la seguente semantica:

$$concurrent^{\mathcal{I}}(x) = \{y : Static \sqcup Event \mid y =_t x\}^{\mathcal{I}};$$

$$precedent^{\mathcal{I}}(x) = \{y : Static \sqcup Event \mid y < x\}^{\mathcal{I}};$$

$$subsequent^{\mathcal{I}}(x) = \{y : Static \sqcup Event \mid y > x\}^{\mathcal{I}}.$$

Conviene definire anche una costante reale, per assegnare un valore di 'istante temporale' anche agli individui statici. Gli individui statici possono essere visti come sempre 'attuali', 'presenti', per

[2] Suppongo predefiniti gli insiemi *Real*, dei numeri reali, e *Rational*, dei numeri razionali.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

cui chiamiamo *_now_* il loro tempo. Le linee universo *ULine*, invece, rappresentano il se' di ogni individuo dinamico e dunque non sono calate nel tempo: ad essi potremmo attribuire una costante di tempo fittizia (e chiamarla *_self_* , ad esempio), oppure nessun valore temporale. Scegliamo la seconda via.

$$\_now\_ \in Real$$

Ovviamente sarà l'interpretazione ad assegnare i valori alle costanti. In particolare, se dovesse essere stata introdotta la costante *_self_*, essa dovrà essere scelta in modo tale da risultare effettivamente un valore 'fittizio', atto a non ingenerare confusione con gli istanti temporali effettivi. Invece *_now_* rappresenta per l'interpretazione l'istante 'presente', il discrimine tra il 'passato' e il 'futuro', che discuteremo più avanti.

L'insieme degli eventi *Event* risulta (parzialmente) ordinato. Se consideriamo una singola linea universo $\hat{x} \in ULine$ e l'insieme X degli eventi ad essa associati,

$$X = event(\hat{x}) \equiv event|_{\hat{x}} = Event \sqcap \exists uline.\{\hat{x}\}^{3} \qquad 3\text{-}34$$

allora X risulta totalmente ordinato ed è rappresentabile con una linea orientata, con l'orientamento indotto da *Time* (Figura 3-10).

Postulo che ogni individuo statico abbia il suo tempo uguale a *_now_*:

$$Static \sqsubseteq \exists time.\{\_now\_\} \qquad 3\text{-}35$$

E' utile definire il concetto *Now* degli individui 'attuali', aventi il valore di *time* uguale a *_now_:*

$$Now \equiv \exists time.\{\_now\_\} \qquad 3\text{-}36$$

e il ruolo *now* che associa ogni *ULine* all'*Event* 'attuale' corrispondente e ogni *Static* a se stesso:

$$now: Static \sqcup ULine \rightarrow Now \qquad 3\text{-}37$$

secondo la seguente corrispondenza:

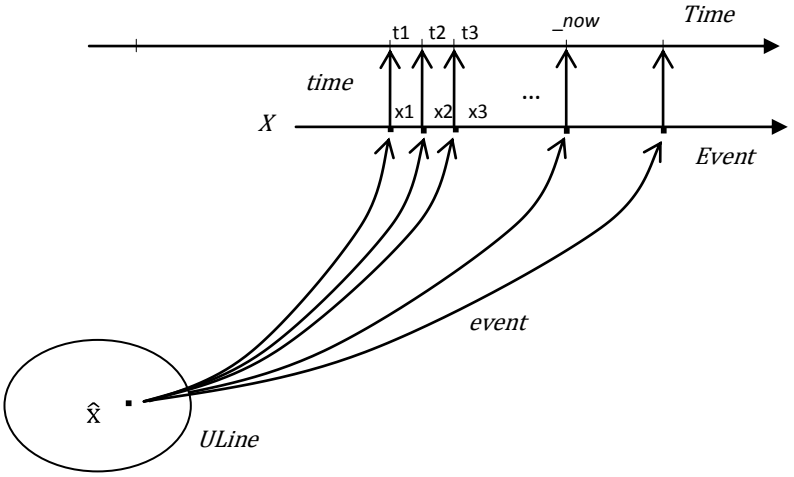

Figura 3-10 *Event* è orientato

[3] Convenzione 2: da qui in avanti, adotto la convenzione per cui, dato un ruolo r, le due notazioni $r(x)$ oppure $r|_x$ indicano equivalentemente la proiezione di r su x, ovvero l'insieme $r(x) = r|_x = \{y \mid (x,y) \in r\}$.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$$now|_x = \text{if } (x \in Static) \text{ then } \{x\} \text{ elsif } (x \in ULine) \text{ then } event|_x \sqcap Now \qquad 3\text{-}38$$

Con questa Ontologia, può definirsi una nuova semantica per le regole sintattiche già esistenti, che si mantengono, ed, in più, si possono definire nuovi concetti, con semantica tipicamente 'temporale'.

3.2.1.1 *Sintassi e Semantica*

Oltre ai concetti ordinari (statici) $\mathcal{SHOIN}$(D), introduciamo concetti dinamici (temporali).

Concetti: $C \rightarrow C_S \mid C_T$

Concetti Statici: $C_S \rightarrow \top \mid \bot \mid A_S \mid C_S \sqcap D_S \mid C_S \sqcup D_S \mid \neg C_S \mid \forall R.C_S \mid \exists R.C_S \mid \{a\} \mid \geq nR \mid \leq nR$

I concetti dinamici (temporali) sono ottenuti da altri concetti tramite operatori temporali.

Concetti Dinamici (Temporali): $C_T \rightarrow C\,\mathcal{U}^{+}D \mid C\,\mathcal{U}^{-}D \mid C\,\mathcal{S}^{+}D \mid C\,\mathcal{S}^{-}D \mid \Diamond^{-}C \mid \Diamond^{+}C \mid \Box^{-}C \mid \Box^{+}C \mid \oplus C \mid \ominus C \mid SubsNextC \mid SubsPrevC \mid \nearrow C \mid \searrow C \mid \rightleftharpoons C \mid C_T \sqcap D \mid C \sqcap D_T \mid C_T \sqcup D \mid C \sqcup D_T \mid \neg C_T$

La semantica dei concetti statici viene estesa in modo naturale in un universo dinamico.

Definisco $\mathcal{I}\_static$ un'interpretazione della *Knowledge Base* $\mathcal{K}$ priva di dimensione temporale. Ogni interpretazione $\mathcal{I}$ nel linguaggio esteso $\mathcal{SHOINT}$ (D) è da intendersi come estensione di una corrispondente $\mathcal{I}\_static$ in $\mathcal{SHOIN}$(D). Per un generico concetto statico C_S, definisco allora la sua interpretazione:

$$C_S^{\mathcal{I}}(x) = C_S^{\mathcal{I}-static}(x), \text{ per } x \in Static \sqcup Dynamic \qquad 3\text{-}39$$

3.2.1.1.1 Ereditarietà: i concetti statici

È necessario aggiungere un'importante proprietà. Relativamente ai soli concetti statici, gli *Event* si comportano, se teniamo un'ottica Object Oriented, come sottoclassi delle *ULine*, nel senso che ogni *Event* eredita l'appartenenza ad un concetto statico della propria *ULine*. Se un concetto statico C è definito sulle *ULine*, esso è definito anche sugli *Event*, ed ogni *Event* eredita il valore d'appartenenza a C dalla propria *ULine*.

$$\text{If } (C : C_s^{\,4}) \text{ then}$$

$$\text{foreach } x : ULine, \text{ foreach } x_i : X = event(x).(\text{If } C^{\mathcal{I}}(x) > 0 \text{ then } C^{\mathcal{I}}(x_i) = C^{\mathcal{I}}(x)). \qquad 3\text{-}40$$

Ad esempio, se *Persona* è il concetto (statico) delle persone e se io sono *Persona*, lo sono per tutta la mia esistenza. È dunque ragionevole attribuire alla mia Linea Universo il concetto di

[4] C_s e C_T sono, propriamente, metaconcetti.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Persona. Ogni mio evento, che è manifestazione nel tempo della mia Linea Universo, è ugualmente *Persona*.

Naturalmente, non è vero il viceversa. Esistono molti concetti statici che sono propri solo degli *Event*, perché sono legati al particolare istante temporale in cui l'evento si cala, e non si trasmettono alla Linea Universo corrispondente. Ad esempio, se *Felice* è il concetto (statico) degli individui felici e se io sono *Felice*, lo sono ora, ma non necessariamente questa è una mia caratteristica di tutta la mia esistenza, cioè della mia *ULine*.

Scomodando Aristotele, egli potrebbe denominare *Sostanza* (οὐσία) ogni concetto statico assegnato alle *ULine*, che costituisce la loro essenza necessaria, e *Accidente* (συμβεβηκός) ogni concetto assegnato agli *Event* (3).

Dunque, l'assegnazione di un concetto statico ad una *ULine* si trasmette automaticamente a tutti i suoi *Event* e non viceversa, per cui assegneremo un concetto statico ad una *ULine* quando è una caratteristica imperitura dell'individuo, mentre lo assegneremo ad un *Event* quando è una caratteristica dell'individuo che dipenda dal particolare istante temporale.

□

Riguardo alla semantica dei concetti temporali, $C_{\mathrm{T}}^{\mathcal{I}}(\mathrm{x})$, abbiamo già incontrato nel Capitolo 2, sullo stato dell'arte, alcuni operatori temporali. Applico ed estendo la loro semantica all'universo dinamico appena definito, mantenendo il seguente criterio generale.

Se x è *Static*, si valuta $C^{\mathcal{I}}(\mathrm{x})$ sull'istante, sempre *attuale*, degli *Static, time* = *_now_*; se invece x è una linea universo *ULine*, si esegue una valutazione sull'insieme di tutti i suoi *Event*, dal passato remoto fino, al massimo, al tempo attuale globale *time* = *_now_* − nel caso di operatori sul Passato −, oppure da *time* = *_now_* fino al futuro remoto − nel caso di operatori sul Futuro −; se infine x è un evento *Event*, si valuta sull'insieme di tutti gli eventi corrispondenti alla stessa linea universo, dal passato remoto fino al massimo al 'suo' tempo attuale *time*(x) − nel caso di operatori sul Passato −, oppure da *time* = *time*(x) fino al futuro remoto − nel caso di operatori sul Futuro −.

Ricordo l'assunzione fatta per cui l'insieme *Time* è discreto e limitato: le nostre valutazioni terminano certamente.

Until in the past: *Until-past* $C\ \mathcal{U}^{-}D$.

$(C\ \mathcal{U}^{-}D)^{\mathcal{I}}(\mathrm{x}) =$

$$\begin{cases} (C \sqcap D)^{\mathcal{I}}(\mathrm{x}) = \min\left[C^{\mathcal{I}}(\mathrm{x}), D^{\mathcal{I}}(\mathrm{x})\right]; & \text{per x} \in \mathit{Static} \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event(x)} \\ \wedge \mathrm{x}_1 \leq \mathrm{now(x)}}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \sup_{\substack{\mathrm{x}_2 \in \mathrm{event(x)} \\ \wedge \mathrm{x}_2 \leq \mathrm{x}_1}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in \mathit{ULine} \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event(uline(x))} \\ \wedge \mathrm{x}_1 \leq \mathrm{x}}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \sup_{\substack{\mathrm{x}_2 \in \mathrm{event(uline(x))} \\ \wedge \mathrm{x}_2 \leq \mathrm{x}_1}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in \mathit{Event} \end{cases} \qquad \text{3-41}$$

Il concetto $C\ \mathcal{U}^{-}D$ corrisponde all'espressione linguistica naturale 'nel passato di x è esistito un istante in cui x è stato D e prima di questo istante (è esistito un istante in cui) x è stato C' (Figura 3-11).

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

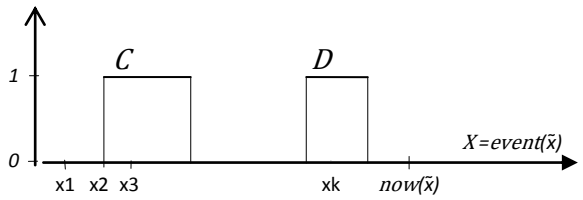

Figura 3-11 l'operatore *Until-past* : $C\,\mathcal{U}^{-}D$, crisp

Ad esempio, il concetto (crisp) *Svuotato* si può esprimere con i concetti (crisp) *Pieno* e *Vuoto* come:

$$Svuotato \equiv Pieno\,\mathcal{U}^{-}\,Vuoto$$

Un altro esempio è il concetto di *Spento*, nel senso di passato da *On* a *Off*:

$$SwitchedOff \equiv On\,\mathcal{U}^{-}Off$$

Nel passaggio ad un ambiente fuzzy, vedremo che sarà possibile esprimere concetti tipicamente fuzzy come *Arricchito*, *Riscaldato*, *Dimagrito* ecc.

Until in the future: *Until-fut* $C\,\mathcal{U}^{+}D$.

Simmetricamente ad *Until-past* $C\,\mathcal{U}^{-}D$,

$$(C\,\mathcal{U}^{+}D)^{\mathcal{I}}(\mathrm{x}) =$$

$$\begin{cases} (C \sqcap D)^{\mathcal{I}}(\mathrm{x}) = \min\left[C^{\mathcal{I}}(\mathrm{x}), D^{\mathcal{I}}(\mathrm{x})\right]; & \text{per x} \in Static \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event(x)} \\ \wedge \mathrm{x}_1 \geq \mathrm{now(x)}}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \sup_{\substack{\mathrm{x}_2 \in \mathrm{event(x)} \\ \wedge \mathrm{now(x)} \leq \mathrm{x}_2 \leq \mathrm{x}_1}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in ULine \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event(uline(x))} \\ \wedge \mathrm{x}_1 \geq \mathrm{x}}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \sup_{\substack{\mathrm{x}_2 \in \mathrm{event(uline(x))} \\ \wedge \mathrm{x} \leq \mathrm{x}_2 \leq \mathrm{x}_1}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in Event \end{cases} \qquad 3\text{-}42$$

che corrisponde all'espressione linguistica naturale 'nel futuro di x esiste un istante in cui x è D ed esiste un istante, precedente ad esso e successivo all'istante attuale, in cui x è C'.

Si possono dare delle versioni alternative dell'operatore *Until* $\mathcal{U}^{\pm}$:

- chiamare quello definito dalle 3-18 e 3-19 *Until inclusivo* $\mathcal{U}_{[\ ]}^{\pm}$, poiché prende in considerazione intervalli temporali con gli estremi compresi; definire l'operatore *Until esclusivo* $\mathcal{U}_{(\ )}^{\pm}$, i cui intervalli temporali non comprendano gli estremi: di conseguenza, nella semantica tutti i segni '$\leq$' sono da sostituire con '$<$' ed i segni '$\geq$' con '$>$'. Inoltre, per x ∈ *Static*, la semantica avrà il valore 0 sempre.
- chiamare quello definito dalle 3-18 e 3-19 *Until (inclusivo) debole* $\tilde{\mathcal{U}}_{[\,]}^{\pm}$, poiché 'si accontenta' che *esista un* istante x_2 nell'intervallo $[\mathrm{x}_1,\mathrm{x}]$ o $[\mathrm{x},\mathrm{x}_1]$ in cui valga C (x_2). Possiamo invece pretendere che C valga su *tutto* l'intervallo e otteniamo l'operatore *Until (inclusivo) strong* $\bar{\mathcal{U}}_{[\,]}^{\pm}$:

$$(C\,\bar{\mathcal{U}}_{[\,]}^{-}D)^{\mathcal{I}}(\mathrm{x}) =$$

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$$\begin{cases} (\mathrm{C} \sqcap \mathrm{D})^{\mathcal{I}}(\mathrm{x}) = \min\left[\mathrm{C}^{\mathcal{I}}(\mathrm{x}), \mathrm{D}^{\mathcal{I}}(\mathrm{x})\right]; & \text{per x} \in Static \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event}(\mathrm{x}) \\ \wedge \mathrm{x}_1 \leq \mathrm{now}(\mathrm{x})}} \{\min\left[\mathrm{D}^{\mathcal{I}}(\mathrm{x}_1), \inf_{\substack{\mathrm{x}_2 \in \mathrm{event}(\mathrm{x}) \\ \wedge \mathrm{x}_2 \leq \mathrm{x}_1}} \mathrm{C}^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in ULine \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event}(\mathrm{uline}(\mathrm{x})) \\ \wedge \mathrm{x}_1 \leq \mathrm{x}}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \inf_{\substack{\mathrm{x}_2 \in \mathrm{event}(\mathrm{uline}(\mathrm{x})) \\ \wedge \mathrm{x}_2 \leq \mathrm{x}_1}} \mathrm{C}^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in Event \end{cases} \quad \text{3-43}$$

$(C\ \bar{\mathcal{U}}_{[\,]}^{+} D)^{\mathcal{I}}(\mathrm{x}) =$

$$\begin{cases} (C \sqcap D)^{\mathcal{I}}(\mathrm{x}) = \min\left[C^{\mathcal{I}}(\mathrm{x}), D^{\mathcal{I}}(\mathrm{x})\right]; & \text{per x} \in Static \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event}(\mathrm{x}) \\ \wedge \mathrm{x}_1 \geq \mathrm{now}(\mathrm{x})}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \inf_{\substack{\mathrm{x}_2 \in \mathrm{event}(\mathrm{x}) \\ \wedge \mathrm{now}(\mathrm{x}) \leq \mathrm{x}_2 \leq \mathrm{x}_1}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in ULine \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event}(\mathrm{uline}(\mathrm{x})) \\ \wedge \mathrm{x}_1 \geq \mathrm{x}}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \inf_{\substack{\mathrm{x}_2 \in \mathrm{event}(\mathrm{uline}(\mathrm{x})) \\ \wedge \mathrm{x} \leq \mathrm{x}_2 \leq \mathrm{x}_1}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in Event \end{cases} \quad \text{3-44}$$

Ovviamente, nel caso crisp e varrà anche per l'estensione fuzzy, nelle espressioni (3.20, 3.21) inf() = 0 se esiste almeno un x_2 per cui $C(\mathrm{x}_2) = 0$. Se tale x_2 non esiste, nel caso crisp per ogni x_2 sarà $C(\mathrm{x}_2) = 1$ ed allora $\inf() = 1$.

La semantica dell'operatore *Until - strong* è forse più aderente al significato verbale della parola 'until' nel linguaggio parlato comune: 'x è prima o poi D e, finché non è D, è sempre C', nel passato o nel futuro, mentre *Until - weak* corrisponde ad un doppio 'prima o poi': 'x è prima o poi D e, finché non è D, è prima o poi C'.

Since in the past: *Since-past* $C\ \mathcal{S}^{-} D$.

$(C\ \mathcal{S}^{-} D)^{\mathcal{I}}(\mathrm{x}) =$

$$\begin{cases} (C \sqcap D)^{\mathcal{I}}(\mathrm{x}) = \min\left[C^{\mathcal{I}}(\mathrm{x}), D^{\mathcal{I}}(\mathrm{x})\right]; & \text{per x} \in Static \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event}(\mathrm{x}) \\ \wedge \mathrm{x}_1 \leq \mathrm{now}(\mathrm{x})}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \sup_{\substack{\mathrm{x}_2 \in \mathrm{event}(\mathrm{x}) \\ \wedge \mathrm{x}_1 \leq \mathrm{x}_2 \leq \mathrm{now}(\mathrm{x})}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in ULine \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event}(\mathrm{uline}(\mathrm{x})) \\ \wedge \mathrm{x}_1 \leq \mathrm{x}}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \sup_{\substack{\mathrm{x}_2 \in \mathrm{event}(\mathrm{uline}(\mathrm{x})) \\ \wedge \mathrm{x}_1 \leq \mathrm{x}_2 \leq \mathrm{x}}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in Event \end{cases} \quad \text{3-45}$$

Il concetto $C\ \mathcal{S}^{-} D$ corrisponde all'espressione linguistica naturale 'nel passato di x è esistito un istante in cui x è stato D e da allora in poi, fino ad adesso, (è esistito un istante in cui) x è stato C'. Esempi sono *VisibileDaQuandoAcceso*: *Visible* $\mathcal{S}^{-}$ *On*, *RottoDaQuandoCaduto*: *Broken* $\mathcal{S}^{-}$ *FallenDown*.

Since in the future: *Since-fut* $C\ \mathcal{S}^{+} D$.

$(C\ \mathcal{S}^{+} D)^{\mathcal{I}}(\mathrm{x}) =$

$$\begin{cases} (C \sqcap D)^{\mathcal{I}}(\mathrm{x}) = \min\left[C^{\mathcal{I}}(\mathrm{x}), D^{\mathcal{I}}(\mathrm{x})\right]; & \text{per x} \in Static \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event}(\mathrm{x}) \\ \wedge \mathrm{x}_1 \geq \mathrm{now}(\mathrm{x})}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \sup_{\substack{\mathrm{x}_2 \in \mathrm{event}(\mathrm{x}) \\ \wedge \mathrm{x}_2 \geq \mathrm{x}_1}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in ULine \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event}(\mathrm{uline}(\mathrm{x})) \\ \wedge \mathrm{x}_1 \geq \mathrm{x}}} \{\min\left[D^{\mathcal{I}}(\mathrm{x}_1), \sup_{\substack{\mathrm{x}_2 \in \mathrm{event}(\mathrm{uline}(\mathrm{x})) \\ \wedge \mathrm{x}_2 \geq \mathrm{x}_1}} C^{\mathcal{I}}(\mathrm{x}_2)\right]\}; & \text{per x} \in Event \end{cases} \quad \text{3-46}$$

che corrisponde all'espressione linguistica 'nel futuro di x esiste un istante in cui x è D e, da allora in poi, (esiste un istante in cui) x è C'.

Come si vede, gli operatori *Until* e *Since* hanno semantica complementare, nel senso che scandiscono l'asse temporale dividendolo in quattro parti, secondo lo schema di Figura 3-12.

Sometime in the past: *Some-past* $\Diamond^{-}C$

$$(\Diamond^{-}C)^{\mathcal{I}}(\mathrm{x}) = \begin{cases} C^{\mathcal{I}}(\mathrm{x}); & \text{per x} \in Static \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event(x)} \\ \wedge \mathrm{x}_1 \le \mathrm{now(x)}}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per x} \in ULine \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event(uline(x))} \\ \wedge \mathrm{x}_1 \le \mathrm{x}}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per x} \in Event \end{cases} \qquad 3\text{-}47$$

Sometime in the future: *Some-future* $\Diamond^{+}C$

$$(\Diamond^{+}C)^{\mathcal{I}}(\mathrm{x}) = \begin{cases} C^{\mathcal{I}}(\mathrm{x}); & \text{per x} \in Static \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event(x)} \\ \wedge \mathrm{x}_1 \ge \mathrm{now(x)}}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per x} \in ULine \\ \sup_{\substack{\mathrm{x}_1 \in \mathrm{event(uline(x))} \\ \wedge \mathrm{x}_1 \ge \mathrm{x}}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per x} \in Event \end{cases} \qquad 3\text{-}48$$

Some-past e *Some-future* corrispondono alle espressioni linguistiche, rispettivamente, 'x è stato (qualche volta) *C'* e 'x sarà (qualche volta) *C'*.

Ad esempio, si può definire il concetto di cliente come colui che ha compiuto almeno un acquisto nel suo passato, o di mortale come di colui che prima o poi sarà morto:

$$Customer \sqsubseteq \Diamond^{-}\exists buys.\top \; ; \; Mortal \sqsubseteq \Diamond^{+}Died$$

Always in the past: *All-past* $\Box^{-}C$

$$(\Box^{-}C)^{\mathcal{I}}(\mathrm{x}) = \begin{cases} C^{\mathcal{I}}(\mathrm{x}); & \text{per x} \in Static \\ \inf_{\substack{\mathrm{x}_1 \in \mathrm{event(x)} \\ \wedge \mathrm{x}_1 \le \mathrm{now(x)}}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per x} \in ULine \\ \inf_{\substack{\mathrm{x}_1 \in \mathrm{event(uline(x))} \\ \wedge \mathrm{x}_1 \le \mathrm{x}}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per x} \in Event \end{cases} \qquad 3\text{-}49$$

Always in the future: *All-future* $\Box^{+}C$

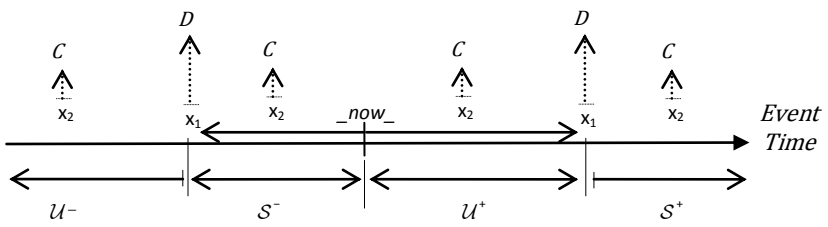

Figura 3-12 schema degli operatori *Until* e *Since*

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$$(\Box^{+}C)^{\mathcal{I}}(\mathrm{x}) = \begin{cases} C^{\mathcal{I}}(\mathrm{x}); & \text{per x} \in Static \\ \inf_{\substack{\mathrm{x}_1 \in \mathrm{event(x)} \\ \wedge \mathrm{x}_1 \geq \mathrm{now(x)}}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per x} \in ULine \\ \inf_{\substack{\mathrm{x}_1 \in \mathrm{event(uline(x))} \\ \wedge \mathrm{x}_1 \geq \mathrm{x}}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per x} \in Event \end{cases} \qquad 3\text{-}50$$

All-past e *All-future* corrispondono alle espressioni linguistiche in linguaggio naturale, rispettivamente, 'x è stato sempre *C*' e 'x sarà sempre *C*'.

Anche per questi ultimi quattro operatori $\Diamond^{\pm}C$ e $\Box^{\pm}C$ sono ammesse le definizioni esclusive, sostituendo i segni '≤' e '≥' rispettivamente con '<' e '>' – per x appartenente ad *ULine* o *Event* – e assegnando invece semantica 0 per x appartenente a *Static*.

Ricordando nuovamente che *Time* è discreto e limitato, dunque finito, in tutte le definizioni della semantica il sup si riduce effettivamente ad un max, mentre l'inf si riduce ad un min.

Ricordando inoltre le norme scelte per l'unione e l'intersezione, valide anche nel crisp, si può facilmente vedere, dalla corrispondenza con formule FOL descritta nella Tabella 3-1, il significato e

Tabella 3-1 corrispondenza tra la semantica degli operatori temporali e formule FOL

$\Diamond^{-}C(\mathrm{x}) \leftrightarrow \sqcup_{x_1 \leq x} C(x_1) \overset{FOL}{\leftrightarrow} \vee_{x_1 \leq x} C(x_1)$
$\Diamond^{+}C(\mathrm{x}) \leftrightarrow \sqcup_{x_1 \geq x} C(x_1) \overset{FOL}{\leftrightarrow} \vee_{x_1 \geq x} C(x_1)$
$\Box^{-}C(\mathrm{x}) \leftrightarrow \sqcap_{x_1 \leq x} C(x_1) \overset{FOL}{\leftrightarrow} \wedge_{x_1 \leq x} C(x_1)$
$\Box^{+}C(\mathrm{x}) \leftrightarrow \sqcap_{x_1 \geq x} C(x_1) \overset{FOL}{\leftrightarrow} \wedge_{x_1 \geq x} C(x_1)$
$C\,\mathcal{U}^{-}D(\mathrm{x}) \leftrightarrow \Diamond^{-}(D \sqcap \Diamond^{-}C)(\mathrm{x}) \overset{FOL}{\leftrightarrow} \vee_{x_1 \leq x}(D(x_1) \wedge \vee_{x_2 \leq x_1} C(x_2))$
$C\,\bar{\mathcal{U}}^{-}D(\mathrm{x}) \leftrightarrow \Diamond^{-}(D \sqcap \Box^{-}C)(\mathrm{x}) \overset{FOL}{\leftrightarrow} \vee_{x_1 \leq x}(D(x_1) \wedge_{x_2 \leq x_1} C(x_2))$
$C\,\mathcal{U}^{+}D(\mathrm{x}) \overset{FOL}{\leftrightarrow} \vee_{x_1 \geq x}(D(x_1) \wedge \vee_{x \leq x_2 \leq x_1} C(x_2))$
$C\,\bar{\mathcal{U}}^{+}D(\mathrm{x}) \overset{FOL}{\leftrightarrow} \vee_{x_1 \geq x}(D(x_1) \wedge \vee_{x \leq x_2 \leq x_1} C(x_2))$
$C\,\mathcal{S}^{-}D(\mathrm{x}) \overset{FOL}{\leftrightarrow} \vee_{x_1 \leq x}(D(x_1) \wedge \vee_{x_1 \leq x_2 \leq x} C(x_2))$
$C\,\bar{\mathcal{S}}^{-}D(\mathrm{x}) \overset{FOL}{\leftrightarrow} \vee_{x_1 \leq x}(D(x_1) \wedge_{x_1 \leq x_2 \leq x} C(x_2))$
$C\,\mathcal{S}^{+}D(\mathrm{x}) \leftrightarrow \Diamond^{+}(D \sqcap \Diamond^{+}C)(\mathrm{x}) \overset{FOL}{\leftrightarrow} \vee_{x_1 \geq x}(D(x_1) \wedge \vee_{x_2 \geq x_1} C(x_2))$
$C\,\bar{\mathcal{S}}^{+}D(\mathrm{x}) \leftrightarrow \Diamond^{+}(D \sqcap \Box^{+}C)(\mathrm{x}) \overset{FOL}{\leftrightarrow} \vee_{x_1 \geq x}(D(x_1) \wedge_{x_2 \geq x_1} C(x_2))$

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

la giustificazione della semantica proposta.

Nella Tabella 3-1 tutti gli x_i devono corrispondere alla stessa *ULine* e, nella parte destra di ogni formula FOL, x è da sostituire con *_now_* se $x \in$ *ULine.*

È forse da sottolineare che *always* e *sometime* non sono operatori primitivi, bensì si possono ricavare da *Until* e *Since*:

$$\Diamond^{+} C = \top \mathcal{U}^{+} C = \top \mathcal{S}^{+} C \qquad 3\text{-}51$$

$$\Diamond^{-} C = \top \mathcal{U}^{-} C = \top \mathcal{S}^{-} C \qquad 3\text{-}52$$

3.2.1.1.2 Ereditarietà: i concetti temporali

Pare evidente come non possa sussistere alcuna ereditarietà, in generale, per i concetti temporali. Essi sono, per definizione dipendenti dall'istante temporale di valutazione, che può essere l'istante di esistenza per l'*Event* oppure l'istante determinato dalla costante *_now_* per la *ULine*. Quindi, in generale, essi differiscono. Basta riconsiderare gli esempi proposti (*Svuotato*, *Dimagrito*, *Customer* ecc.), per rendersene conto.

Fanno eccezione quei concetti temporali che sono verificati sugli *Event* che si trovano alle estremità della Linea Universo: *firstEvent* per concetti sul passato e *lastEvent* per concetti sul futuro. Ad esempio, $Alive \sqsubseteq \Diamond^{-}\exists\, todayGenerated$ o $Mortal \sqsubseteq \Diamond^{+}\exists\, todayDied$, se i ruoli *todayGenerated* e *todayDied* sono proprietà booleane, che valgono true nel giorno, rispettivamente, di generazione e di morte dell'individuo e, ovviamente se la sequenza degli eventi inizia con la generazione e termina con la morte. Ma, in generale, ribadiamo, non esiste nessuna ereditarietà tra concetti temporali.

□

Abbiamo ipotizzato *Time* limitato; possiamo dunque definire i due numeri reali elementi di *Time* e suoi estremi, tramite i due ruoli booleani *firstTime* e *lastTime:*

$$firstTime : Time \rightarrow Boolean \qquad 3\text{-}53$$

[5]

$$lastTime : Time \rightarrow Boolean \qquad 3\text{-}54$$

$$firstTime^{\mathcal{I}}(x) = \{(x = inf\{y \mid y \in Time\})\} \qquad 3\text{-}55$$

$$lastTime^{\mathcal{I}}(x) = \{(x = sup\{y \mid y \in Time\})\} \qquad 3\text{-}56$$

Se $Time \neq \phi$, sono così univocamente definiti (al più coincidenti, nel caso di unico elemento) gli unici due t_f e $t_l \in$ *Time* per cui $(t_f, true) \in firstTime$ e $(t_l, true) \in LastTime$.

[5] L'insieme *Boolean* = {*true*, *false*} è supposto predefinito.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Consideriamo in *Time* (insieme numerico orientato totalmente) il nuovo ruolo *nextTime*, che associa ad ogni elemento di *Time* il suo successore in *Time* (se esiste), ed il ruolo *prevTime* (*previous time*), che associa ad ogni elemento di *Time* il suo predecessore (se esiste) (Figura 3-13).

$$nextTime : Time \rightarrow Time \quad 3\text{-}57$$

$$prevTime : Time \rightarrow Time \quad 3\text{-}58$$

$$nextTime^{\mathcal{I}} = \{(x, y) \mid x,y \in Time \wedge (x, false) \in lastTime \wedge y^{\mathcal{I}} = inf \{y \mid y \in Time \wedge y > x\}\} \quad 3\text{-}59$$

$$prevTime^{\mathcal{I}} = \{(x, y) \mid x,y \in Time \wedge (x, false) \in firstTime \wedge y^{\mathcal{I}} = sup \{y \mid y \in Time \wedge y < x\}\} \quad 3\text{-}60$$

che restano indefiniti ai due estremi.

Grazie ad essi, possiamo definire i ruoli corrispondenti in *Event, next* e *prev* : definiamo prima *firstEvent* e *lastEvent* come i ruoli che restituiscono il primo evento e l'ultimo di una fissata *ULine*.

$$firstEvent : ULine \rightarrow Event \quad 3\text{-}61$$

$$lastEvent : ULine \rightarrow Event \quad 3\text{-}62$$

$$firstEvent^{\mathcal{I}}(x) = \{ y \mid y \in Event \wedge y^{\mathcal{I}} = inf\{z \mid z \in event(x) \}\}, \; x \in ULine \quad 3\text{-}63$$

$$lastEvent^{\mathcal{I}}(x) = \{ y \mid y \in Event \wedge y^{\mathcal{I}} = sup\{z \mid z \in event(x) \}\}, \; x \in ULine \quad 3\text{-}64$$

Devo prima ricordare che ho descritto *Time* come un insieme discreto, limitato, di punti reali, con distanza non necessariamente costante tra un punto e il suo successore. Aggiungo l'ipotesi che ogni linea universo, durante la sua *'esistenza'*, ovvero nell'intervallo temporale tra il primo e l'ultimo suo evento, abbia uno e un solo evento, per ogni istante in *Time.*

$$\underline{\text{Hyp}}: \forall x \in ULine, \forall t \in Time \cap [(firstEvent.time)(x), (lastEvent.time)(x)] \; \exists! \, y \in X \equiv$$

$$event(x) \mid time(y) = t. \quad 3\text{-}65$$

Allora, *next* sarà definito come il ruolo funzionale che associa un evento al suo successore (se esiste è unico), mentre *prev* è il ruolo funzionale che associa un evento al suo predecessore (se esiste è unico) (Figura 3-13).

$$next : Event \rightarrow Event \quad 3\text{-}66$$

$$prev : Event \rightarrow Event \quad 3\text{-}67$$

$$fun(next); fun(prev) \quad 3\text{-}68$$

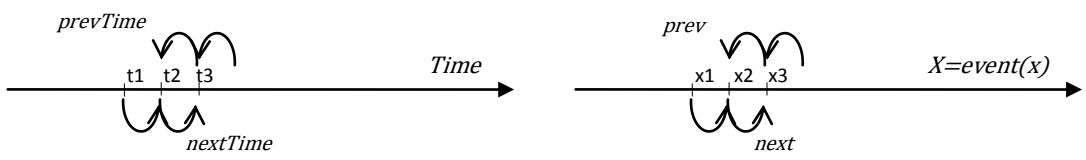

Figura 3-13 *nextTime*, *prevTime*, *next*, *prev.*

$$(next)^{\mathcal{I}}(\mathrm{x}) = \begin{cases} \{\mathrm{y} | \mathrm{y} \in (uline.event)(\mathrm{x}) \wedge time(\mathrm{y}) = (time.nextTime)(\mathrm{x})\}; & \mathrm{x} \neg (uline.lastEvent)(\mathrm{x}) \\ \bot; & \mathrm{x} \in (uline.lastEvent)(\mathrm{x}) \end{cases} \quad 3\text{-}69$$

$$(prev)^{\mathcal{I}}(\mathrm{x}) = \begin{cases} \{\mathrm{y} | \mathrm{y} \in (uline.event)(\mathrm{x}) \wedge time(\mathrm{y}) = (time.prevTime)(\mathrm{x})\}; & \mathrm{x} \neg (uline.firstEvent)(\mathrm{x}) \\ \bot; & \mathrm{x} \in (uline.firstEvent)(\mathrm{x}) \end{cases} \quad 3\text{-}70$$

Possiamo così introdurre due nuovi operatori sui Concetti, *Next* o $\oplus$ e *Prev* o $\ominus$, con la seguente definizione.

Definizione 8. Dato un concetto C, il concetto $NextC \equiv \oplus C$ associa ad ogni individuo x il valore che l'individuo $next(\mathrm{x})$ assume in C:

$$\underline{NextC \equiv \oplus C}$$

$$(\oplus C)^{\mathcal{I}}(\mathrm{x}) = \begin{cases} C^{\mathcal{I}}(\mathrm{x}); & \text{per x} \in Static \\ C^{\mathcal{I}}(next^{\mathcal{I}}(\mathrm{x})); & \text{per x} \in Event \\ C^{\mathcal{I}}(\mathrm{x}); & \text{per x} \in ULine \end{cases} \quad 3\text{-}71$$

L'operatore $\oplus$ applicato al concetto C ha l'effetto di proiettare ogni suo elemento avanti nel tempo di uno step fino al successore. Se *Time* è suddiviso in intervalli regolari da un'unità temporale, si può vedere questa operazione come una traslazione all'indietro dell'asse temporale, esclusivamente per ciò che attiene ai valori di C, della stessa unità temporale (Figura 3-14).

Definizione 9. Dato un concetto C, il concetto $PrevC \equiv \ominus C$[6] associa ad ogni individuo x il valore che l'individuo $prev(\mathrm{x})$ assume in C:

$$\underline{PrevC \equiv \ominus C}$$

$$(\ominus C)^{\mathcal{I}}(\mathrm{x}) = \begin{cases} C^{\mathcal{I}}(\mathrm{x}); & \text{per x} \in Static \\ C^{\mathcal{I}}(prev^{\mathcal{I}}(\mathrm{x})); & \text{per x} \in Event \\ C^{\mathcal{I}}(\mathrm{x}); & \text{per x} \in ULine \end{cases} \quad 3\text{-}72$$

L'operatore $\ominus$ applicato al concetto C ha l'effetto di proiettare ogni suo elemento indietro nel tempo di uno step fino al predecessore o, equivalentemente, trasla in avanti l'asse temporale di un'unità (quando è definita) (Figura 3-15).

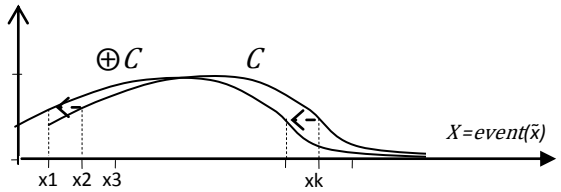

Figura 3-14 *NextC*, in generale

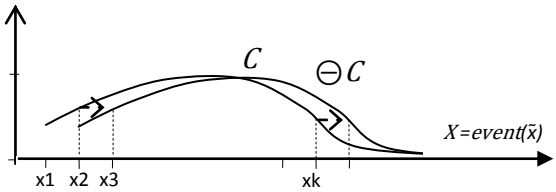

Figura 3-15 *PrevC*, in generale

[6] Preferisco denominare *prev* (previous) l'operatore $\ominus$ anziché *last*, come avrei dovuto fare seguendo la bibliografia, per non ingenerare confusione con *lastTime* o *lastEvent*, che indicano gli *ultimi* tempi o eventi, mentre $\ominus$ indica il *precedente*, non l'ultimo.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Notiamo che $\oplus C$ non è definito se x è il *lastEvent* della sua linea universo; analogamente, $\ominus C$ non è definito se x è il *firstEvent* della sua linea Universo.[7]

Se torniamo alla definizione degli operatori *Until* $\mathcal{U}$ e *Since* $\mathcal{S}$, vediamo facilmente che anche gli operatori $\oplus$ e $\ominus$ non sono primitivi, ma si possono derivare da *Until* e *Since* esclusivi $\mathcal{U}_{()}$ e $\mathcal{S}_{()}$:

$$\oplus C = \bot \, \mathcal{U}_{()}^{+} \, C \qquad 3\text{-}73$$

$$\ominus C = \bot \, \mathcal{S}_{()}^{-} \, C \qquad 3\text{-}74$$

Infatti, $\bot \, \mathcal{U}_{()}^{+} \, C^{\mathcal{I}}(\mathrm{x})$ è ≠0 solo per x_1 = *next*(x) e $\bot \, \mathcal{U}_{()}^{+} \, C^{\mathcal{I}}(\mathrm{x}) = C\,(\mathit{next}(\mathrm{x}))$ e $\bot \, \mathcal{S}_{()}^{-} C^{\mathcal{I}}(\mathrm{x})$ *è* ≠0 solo per x_1 = *prev* (x) e $\bot \, \mathcal{S}_{()}^{-} C^{\mathcal{I}}(\mathrm{x}) = C\,(\mathit{prev}(\mathrm{x}))$.

Grazie agli operatori $\oplus$ e $\ominus$, è anche possibile dare delle definizioni iterative incrementali degli altri operatori temporali. Per gli operatori inclusivi si veda la Tabella 3-2.

Diviene interessante considerare una singola linea universo e gli eventi ad essa corrispondenti. Sia $\tilde{\mathrm{x}} \in \mathit{ULine}$ e $X = \mathit{event}(\tilde{\mathrm{x}})$. Allora $A \sqcap X$ sarà il concetto contenente solo individui di X, che appartengano anche ad A: $A \sqcap X$ è cioè la restrizione di A su X.

Sarà più istruttivo quando ci estenderemo nel fuzzy, ma già in ambito crisp si può vedere come, se applichiamo l'operatore $\oplus$ ad $A \sqcap X$, $\oplus(A \sqcap X)$ diviene il concetto contenente gli individui di X i cui successori siano in A.

Supponiamo che, nella sua esistenza, $\tilde{\mathrm{x}}$ divenga e rimanga A, senza che lo sia stato inizialmente, cioè che l'appartenenza di X in A abbia un andamento a gradino crescente come in Figura 3-16.

Si ha che $A \sqcap X(\mathrm{x_i}) = \begin{cases} 0 \text{ per } \mathrm{i} < k \\ 1 \text{ per } \mathrm{i} \geq \mathrm{k} \end{cases}$ e $\oplus(A \sqcap X)(\mathrm{x_i}) = \begin{cases} 0 \text{ per } \mathrm{i} < k-1 \\ 1 \text{ per } \mathrm{i} \geq \mathrm{k}-1 \end{cases}$.

Allora, per ogni x ∈ X, si ha che $\oplus(A \sqcap X)(\mathrm{x}) = (A \sqcap X)(\mathit{next}(\mathrm{x})) \geq (A \sqcap X)(\mathrm{x}) = A(\mathrm{x})$ e dunque $(A \sqcap X) \subseteq \oplus(A \sqcap X)$ in senso insiemistico e

Tabella 3-2 definizioni iterative incrementali per alcuni operatori temporali inclusivi

$\Diamond^{-} C = C \sqcup \ominus (\Diamond^{-} C)$ $\Diamond^{-} C(\mathrm{x_0}) = C(\mathrm{x_0})$	$\Diamond^{+} C = C \sqcup \oplus (\Diamond^{+} C)$ $\Diamond^{+} C(\mathrm{x_0}) = C(\mathrm{x_0})$
$\Box^{-} C = C \sqcup \ominus (\Box^{-} C)$ $\Box^{-} C(\mathrm{x_0}) = C(\mathrm{x_0})$	$\Box^{+} C = C \sqcup \oplus (\Box^{+} C)$ $\Box^{+} C(\mathrm{x_0}) = C(\mathrm{x_0})$
$C \, \mathcal{U}^{-} D = (D \sqcap \Diamond^{-} C) \sqcup \ominus (C \, \mathcal{U}^{-} D)$ $C \, \mathcal{U}^{-} D(\mathrm{x_0}) = (D \sqcap C)\,(\mathrm{x_0})$	$C \, \mathcal{S}^{+} D = (D \sqcap \Diamond^{+} C) \sqcup \oplus (C \, \mathcal{S}^{+} D)$ $C \, \mathcal{S}^{+} D(\mathrm{x_0}) = (D \sqcap C)\,(\mathrm{x_0})$
per $\mathrm{x_0}$=*firstEvent*(*uline*($\mathrm{x_0}$))	per $\mathrm{x_0}$= *lastEvent* (*uline*($\mathrm{x_0}$))

[7] A(⊥)=⊥ (indefinito) per ogni concetto A

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

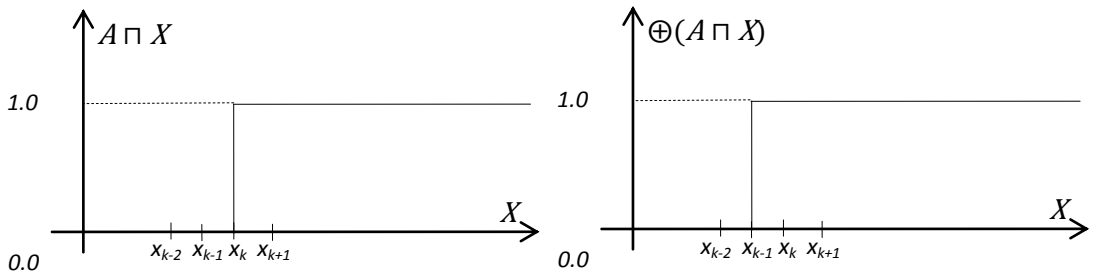

Figura 3-16 andamento a gradino crescente per $A \sqcap X$

$(A \sqcap X) \sqsubseteq \oplus(A \sqcap X)$ come sussunzione di concetti.

Forse è utile sottolineare che qui si considerano i concetti come *insiemi di individui* dell'universo U, per cui ha senso l'operazione $A \sqcap X$, non come membership-function, quand'anche esista esplicita. I concetti concreti hanno membership-function esplicita (sia pur crisp in logica crisp), definita su un dominio concreto (numerico o string) contenuto in Δ_D, ben distinto dall'universo U, per cui non avrebbe senso un'intersezione con un insieme di individui quale è X. Ad esempio, per un sottouniverso di lampadine, si può definire il concetto On con una membership-function, funzione della corrente presente: $On(\mathrm{i}) = \text{if } (\mathrm{i}>0) \text{ then } 1 \text{ else } 0$; quando consideriamo però l'intersezione $On \sqcap X$, questa è l'insieme contenente quegli individui evento x_k di X che hanno valore 1 di On, o se vogliamo quegli individui evento x_k di X, la cui corrente $i(x_k)$ è tale per cui $On(i(x_k))=1$.

Se avessimo un andamento opposto, cioè a gradino ma decrescente, cioè se

$A \sqcap X\ (x_i) = \begin{cases} 1 & \text{per } i < k \\ 0 & \text{per } i \geq k \end{cases}$, allora si otterrebbe che $\oplus(A \sqcap X) \sqsubseteq (A \sqcap X)$ o, equivalentemente, $(A \sqcap X) \sqsubseteq \ominus (A \sqcap X)$.

Si potrebbe tentare di descrivere questa proprietà di crescenza con gli operatori *Until* e *Since strong*, con espressioni come $\neg A\, \bar{\mathcal{U}}\, A$ e $A\, \bar{S}\, A$. Si deve prestare attenzione a due questioni. In primo luogo, uno solo dei due operatori non è sufficiente. Infatti, la semantica di $A\, \bar{\mathcal{U}}\, B$ dice che, finché non accade B, è sempre A, quindi $\neg A\, \bar{\mathcal{U}} A$ potrebbe descrivere un andamento del tipo crescente, ma *Until* nulla dice sugli istanti successivi, ove potrebbe accadere di tutto. Al contrario, la semantica di $A \bar{S} B$ dice che, da quando accade B, è sempre A, ma nulla dice sugli istanti precedenti e nemmeno che l'accadimento di B deve essere il primo. In secondo luogo, neanche la combinazione dei due, scrivendo $\neg A\, \bar{\mathcal{U}} A \sqcap A \bar{S} A$, sarebbe sufficiente, perché anche così rimarrebbe una 'zona d'ombra' intermedia non controllata, ove potrebbero esserci oscillazioni. Una possibile descrizione appropriata con questi operatori dovrebbe utilizzare *Until* debole e la negazione, in un'espressione che esprima decrescenza negata, sia nel passato che nel futuro:

$$((\neg(A\ \mathcal{U}^{-}\neg A)) \sqcap (\neg(A\ \mathcal{U}^{+}\neg A)))(\tilde{x}) \qquad 3\text{-}75$$

per dire che per la linea universo $\tilde{x}$ non esistono istanti di $\neg A$ preceduti da A. Altre espressioni che descrivono crescenza stretta sono le seguenti:

$$(((\neg A\ \bar{\mathcal{U}}^{-}\Box^{+}A) \sqcap \Box^{+}A) \sqcup ((\neg A\ \bar{\mathcal{U}}^{+}\Box^{+}A) \sqcap \Box^{-}\neg A))(\tilde{x}) \qquad 3\text{-}76$$

$$(((A\ \bar{S}^{-}\Box^{-}\neg A) \sqcap \Box^{+}A) \sqcup ((A\ \bar{S}^{+}\Box^{-}\neg A) \sqcap \Box^{-}\neg A))(\tilde{x}) \qquad 3\text{-}77$$

e quindi le loro negazioni esprimono decrescenza non stretta.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Possiamo, comunque, introdurre due operatori sui concetti:

Definizione 10. Si definisce *SubsNext* l'operatore che ad ogni concetto C associa il nuovo concetto $SubsNext(C) \equiv SubsNextC \sqsubseteq ULine$, con la seguente semantica:

$$SubsNextC^{\mathcal{I}}(\tilde{x}) = (CX \sqsubseteq \oplus CX)^{\mathcal{I}},\ CX \equiv C \sqcap X,\ X = event(\tilde{x}),\ \text{per ogni } \tilde{x} \in ULine.$$

Definizione 11. Si definisce *SubsPrev* l'operatore che ad ogni concetto C associa il nuovo concetto $SubsPrev(C) \equiv SubsPrevC \sqsubseteq ULine$, con la seguente semantica:

$$SubsPrevC^{\mathcal{I}}(\tilde{x}) = (CX \sqsubseteq \ominus CX)^{\mathcal{I}},\ CX \equiv C \sqcap X,\ X = event(\tilde{x}),\ \text{per ogni } \tilde{x} \in ULine.$$

Inoltre, affermiamo che

$SubsNextC(\tilde{x}) = C \sqcap X \sqsubseteq \oplus C \sqcap X$ è un indicatore della crescenza (non stretta) e

$SubsPrevC(\tilde{x}) = C \sqcap X \sqsubseteq \ominus C \sqcap X$ è un indicatore della decrescenza (non stretta)

dei valori in C assunti dagli individui eventi appartenenti ad X e corrispondenti alla linea universo $\tilde{x}$.

Osserviamo che, per ogni $\tilde{x} \in ULine$, $SubsNextC^{\mathcal{I}}(\tilde{x}) = SubsPrevC^{\mathcal{I}}(\tilde{x}) = 1$ se e solo se $CX \equiv C \sqcap X$ è costante;

se $CX \neq$ costante allora $SubsNextC^{\mathcal{I}}(\tilde{x}) = \neg\, SubsPrevC^{\mathcal{I}}(\tilde{x})$.

Nelle logiche crisp, se ci limitiamo a valutazione sul solo passato o sul solo futuro, valgono le seguenti equivalenze:

$$SubsNextC^{\mathcal{I}}(\tilde{x}) \leftrightarrow (\neg(A\ \mathcal{U}\neg A))^{\mathcal{I}}(\tilde{x}) \text{ e} \qquad 3\text{-}78$$

$$SubsPrevC^{\mathcal{I}}(\tilde{x}) \leftrightarrow (\neg(\neg A\ \mathcal{U}A))^{\mathcal{I}}(\tilde{x}). \qquad 3\text{-}79$$

Quando passeremo in ambito fuzzy, dove sono possibili crescite graduali nei valori d'appartenenza ad un concetto e dove anche la sussunzione può assumere valori diversi nell'intervallo unitario, le equivalenze non saranno più valide e sarà più appropriato descrivere la proprietà di crescenza e decrescenza con gli operatori *SubsNext* e *SubsPrev,* invece che con l'operatore *Until*. Possiamo comunque dire che l'espressione, ad esempio,

$$SubsNextOn(x) = SubsNext(On)(x) = (On \sqcap X) \sqsubseteq \oplus(On \sqcap X)$$

esprime il fatto che, per (tutti gli eventi del) la linea universo x, x o è sempre spento ($\Box\neg On$: $On(x_i)=0\ \forall x_i \in X$), oppure, da quando si accende (da quando $\exists x_k$, $On(x_k)=1$), x rimarrà acceso per sempre ($On(x_i)=1\ \forall i \geq k$).

Possiamo introdurre allora i nuovi operatori unari '↗' (o '*Increasing*' o '*Incr*') e '↘' (o '*Decreasing*' o '*Decr*'), '⇌' (o '*Constant*' o '*Const*'), che generano i seguenti nuovi concetti.

Definizione 12. Operatori d'andamento (*tendency operators*). Dato il concetto C, gli operatori '↗' ('*Incr*') e '↘' ('*Decr*') e '⇌' ('*Const*'), generano i seguenti nuovi concetti, con la corrispondente semantica.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Increasing: $IncrC \equiv Incr(C) \equiv \nearrow C$.

$$\nearrow C^{\mathcal{I}} \equiv IncrC^{\mathcal{I}} \equiv \{x \mid x \in ULine \wedge SubsNextC(x) \wedge \neg SubsPrevC(x)\}^{\mathcal{I}};$$

Decreasing: $DecrC \equiv Decr(C) \equiv \searrow C$.

$$\searrow C^{\mathcal{I}} \equiv DecrC^{\mathcal{I}} \equiv \{x \mid x \in ULine \wedge SubsPrevC(x) \wedge \neg SubsNextC(x)\}^{\mathcal{I}};$$

Constant: $ConstC \equiv Const(C) \equiv \rightleftharpoons C$.

$$\rightleftharpoons C^{\mathcal{I}} \equiv ConstC^{\mathcal{I}} \equiv \{x \mid x \in ULine \wedge SubsNextC(x) \wedge SubsPrevC(x)\}^{\mathcal{I}}.$$

$\nearrow C$ è cioè il concetto contenente quelle linee universo i cui valori in C sono (strettamente) crescenti, $\searrow C$ è il concetto contenente quelle linee universo i cui valori in C sono (strettamente) decrescenti e $\rightleftharpoons C$ è il concetto contenente quelle linee universo i cui valori in C sono costanti.

Così, per x ∈ *ULine*, ad esempio, se *On* ha il significato di "acceso",

x ∈ $\nearrow$*On* se e solo se "x si è acceso e da allora non si è mai spento".

3.2.1.1.3 Ereditarietà: i ruoli

Analogamente ai concetti statici, introduciamo l'ereditarietà per i ruoli, dalle *ULine* agli *Event*.

I ruoli, in generale, possono essere definiti come aventi sia dominio che range indistintamente *Static* oppure *Dynamic*. Nel caso in cui, nella coppia di individui messi in relazione dal ruolo, uno o entrambi di essi siano *ULine*, allora la stessa relazione si estende su tutti gli *Event* di quella o quelle *ULine*. Ad esempio, se *isMotherOf* è il ruolo che associa una madre ai propri figli, esso si definisce sulle *ULine* e si trasmette sui rispettivi *Event* legati dalla proprietà di contemporaneità (ruolo *concurrent*) (Figura 3-17). I ruoli, invece, definiti tra *Event* o *Static*, poiché dipendenti dal particolare istante temporale, non sono oggetto di alcuna estensione (Figura 3-18).

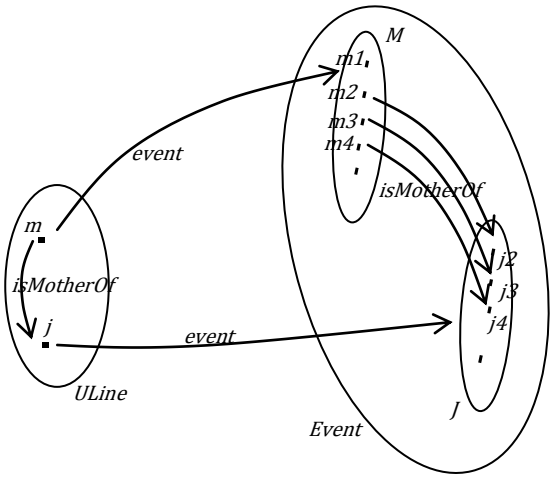

Figura 3-17 esempio di ereditarietà dalle *ULine* agli *Event*, per i ruoli

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

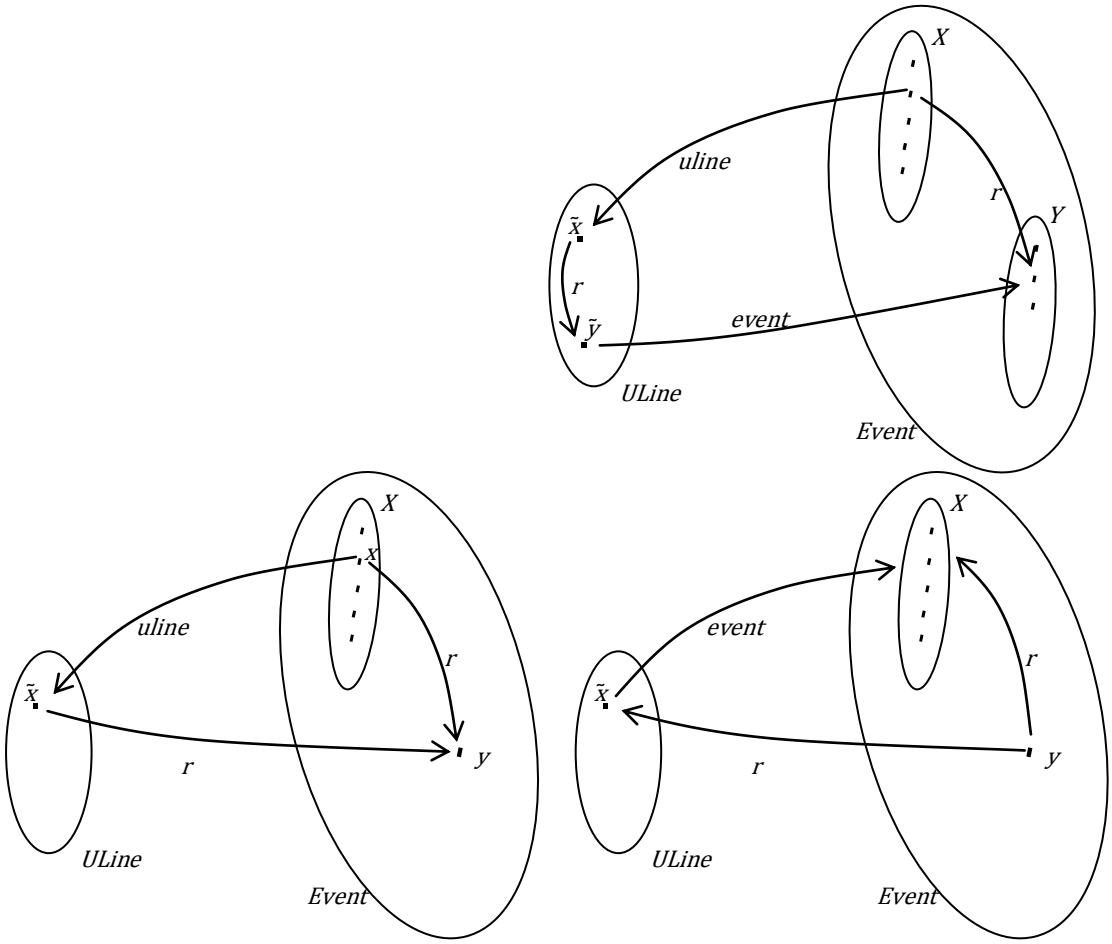

Figura 3-18 ereditarietà per i ruoli, nei tre casi

Definizione 13. Ereditarietà dei ruoli.
If (r: *ULine* → *ULine*) Then (r: *Dynamic* → *Dynamic*) and (r (x) = *event* · r · *uline* (x) ⊓ *concurrent* (x), foreach x: *Event*)

If (r: *ULine* → *Event* ⊔ *Static*) Then (r: *Dynamic* → *Event* ⊔ *Static*) and (r (x) = r · *uline* (x), foreach x: *Event*)

If (r: *Event* ⊔ *Static* → *ULine*) Then (r: *Event* ⊔ *Static* → *Dynamic*) and (r (x) = *event* · r (x) ⊓ *concurrent* (x), foreach x: *Event* ⊔ *Static*).

□

D'ora in poi, quando gli operatori temporali non hanno alcun apice, è inteso che si considera solo il *passato*. Questo è dovuto sia ad esigenze di concisione e sintesi concettuale che alla considerazione che, nella maggior parte dei sistemi concreti, sono analizzati e valutati dati acquisiti, che riguardano il passato. Quando è necessaria, l'estensione alla versione sul futuro degli operatori temporali è comunque immediata.

Esempio 1. Negozio di moda.

In un negozio di abbigliamento, si vuole riconoscere un premio ai clienti il cui numero d'acquisti in un giorno è in crescita, in particolare ai clienti che da quando hanno compiuto più di 3 acquisti,

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

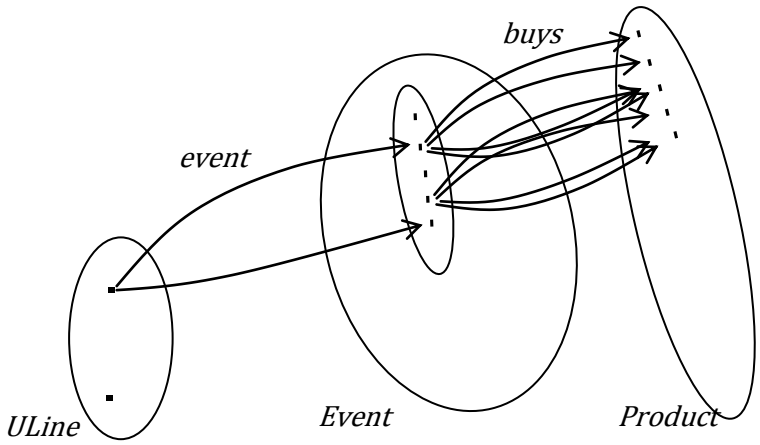

Figura 3-19 *Customer* per l'esempio 1.

non sono mai scesi sotto questo numero. Una Persona è una Linea Universo, concretizzato dall'insieme dei suoi Eventi, uno per ogni giorno significativo. Cliente è una Persona che ha compiuto qualche volta almeno un acquisto (Figura 3-19). Ogni giorno un cliente può compiere più di un acquisto.

$KB \supseteq \{ buys : Event \rightarrow Product , Customer \equiv Person \sqcap \Diamond\exists buys.\top \}$. Inoltre, introduciamo le seguenti definizioni.

$GoodCustomer = Customer \sqcap \geq 3 buys$ è il concetto insieme degli individui evento (istanze nel tempo di clienti) che hanno compiuto più di 3 acquisti.

$SometimeGoodCustomer = ULine \sqcap \Diamond GoodCustomer$ è il concetto insieme dei clienti (linee universo) che hanno compiuto almeno una volta più di 3 acquisti.

$IncreasingGoodCustomer = \nearrow(GoodCustomer)$ è il concetto insieme dei clienti (linee universo) che hanno raggiunto la quota di 3 acquisti e da allora non l'hanno mai lasciata.

$AwardedCustomer = IncreasingGoodCustomer$ è il concetto insieme dei clienti premiati.

Esempio 2. Clienti di banca virtuosi.

Supponiamo che una banca voglia tener traccia dei clienti il cui deposito sia abbastanza stabile, in particolare riferito ad una soglia (500 mila €), oltre la quale il deposito è definito 'gold'. Si vogliono memorizzare i clienti che hanno superato la soglia 'gold' e ne rimangono sempre al di sopra (i clienti 'virtuosi') e quelli che sono sempre stati sopra la soglia (i clienti 'gold'). Ogni cliente è una persona che possiede un deposito bancario. Egli è una linea Universo, che ha, per ogni istante di tempo significativo (ad es., ogni giorno), un proprio evento, rappresentante il cliente fisico in quell'istante.

La *KB* della banca contenga i seguenti concetti e ruoli:

$KB \supseteq \{ Deposit \sqsubseteq Rational$; $hasDeposit : Event \rightarrow Deposit$; $fun(hasDeposit)$; $Customer \equiv Person \sqcap \Diamond\exists hasDeposit.\top$; $GoldDeposit = Deposit \sqcap (\geq 500)$; $\geq 500 \equiv Trap(500, 500, \max, \max, 0, \max)\}$ (Figura 3-20, Figura 3-21).

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

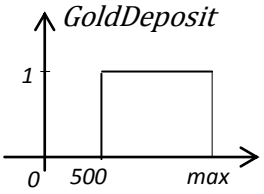

Figura 3-20 *GoldDeposit,* crisp.

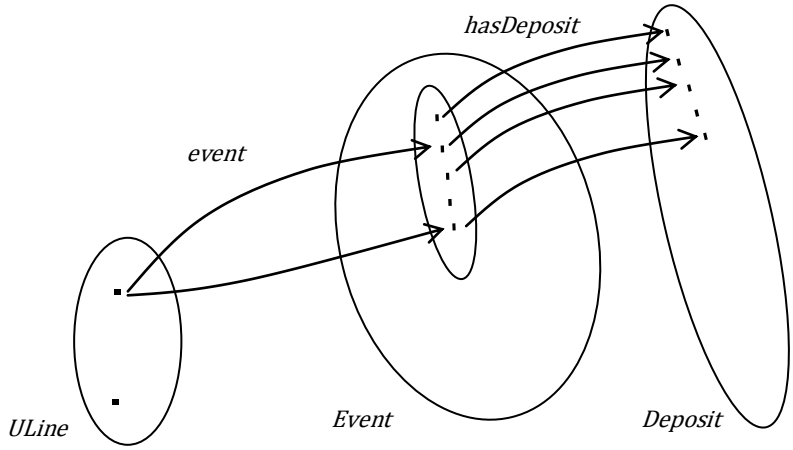

Figura 3-21 *Customer,* per l'esempio 2

Inoltre, introduciamo le seguenti definizioni.

GoldDepCustomer = *Customer* ⊓ ∃*hasDeposit.GoldDeposit* è il concetto insieme degli individui evento (istanze nel tempo di clienti) aventi deposito bancario 'gold'.

SometimeGoldDeposit = *ULine* ⊓ ◇*GoldDepCustomer* è il concetto insieme dei clienti (linee universo) che hanno avuto almeno una volta un deposito 'gold'.

VirtuousCustomer = ↗(*GoldDepCustomer*) è il concetto insieme dei clienti (linee universo) il cui deposito ha superato la soglia 'gold' e ne rimane sempre al di sopra.

GoldCustomer = *SometimeGoldDeposit* ⊓ ⇌(*GoldDepCustomer*) è il concetto insieme dei clienti linee universo che sono sempre stati sopra la soglia 'gold'.

Esempio 3. Agenti vicini al bersaglio.

Quest'esempio sarà propagato lungo le diverse fasi della discussione di questa tesi e progressivamente esteso con i nuovi costrutti. Una parte della progettazione ed implementazione sarà ad esso dedicata.

Si supponga di avere degli agenti automi, aventi ciascuno un bersaglio (anche comune) da raggiungere. Gli agenti sono Linee Universo in movimento, i bersagli sono, in prima approssimazione, Statici. Il sistema può misurare, in ogni istante di tempo significativo, la distanza tra l'agente (il suo evento) ed il corrispondente bersaglio. Il sistema è interessato a registrare quali agenti si avvicinano e rimangono sempre vicini al bersaglio (ad esempio supponendo 'vicina' una distanza di meno di 50 metri). (Figura 3-22, Figura 3-23)

Il *KB* contiene i seguenti concetti e ruoli.

KB ⊇ {*Agent* ⊑ *Dynamic*; *hasDistance* : *Event* → *Distance*; *Distance* ⊑ *PositiveRational*; *CloseDistance* = *Distance* ⊓ (≤50) ; ≤50 ≡ *Trap* (0, 0, 50, 50, 0, 500)}. Inoltre, introduciamo le seguenti definizioni.

CloseAgent = *Agent* ⊓ ∃*hasDistance. CloseDistance* è il concetto insieme degli individui evento (istanze nel tempo di agenti) aventi distanza dal target 'vicina'.

verso $f\_\mathcal{SH}(\mathcal{O}\mathcal{I}\mathcal{Q}^{+}\mathcal{T}$ (D)

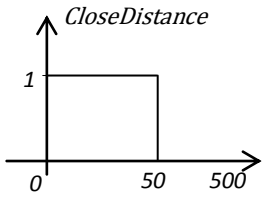

Figura 3-22 *CloseDistance*, crisp

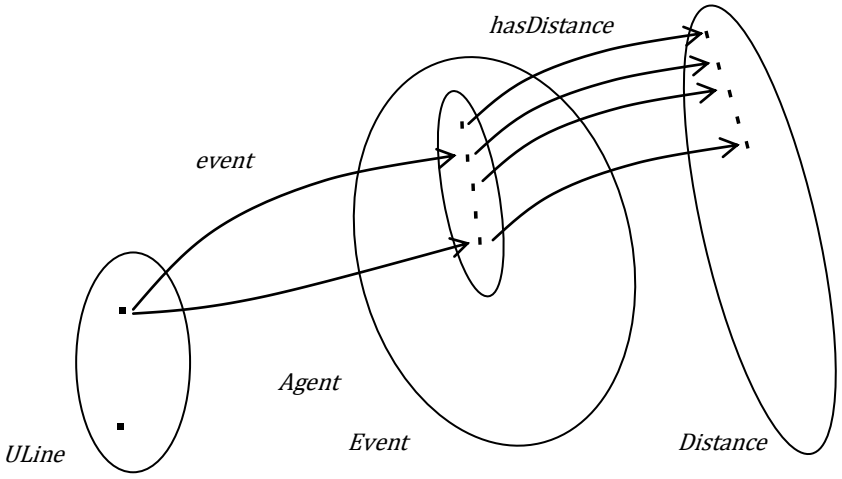

Figura 3-23 *Agent*, per l'esempio 3

$SometimeCloseAgent = ULine \sqcap \Diamond CloseAgent$ è il concetto insieme degli agenti (linee universo) che sono stati almeno una volta a distanza dal target 'vicina'.

$ApproachingAgent = \nearrow(CloseAgent)$ è il concetto insieme degli agenti (linee universo) che si sono avvicinati e non si sono sensibilmente riallontanati.

Esempio 4. Località calde.

In un'agenzia turistica o in una stazione meteorologica o in un ente di rilevamento del clima, siamo interessati a reperire le località aventi temperatura costantemente, o sempre più, calda. Conveniamo sia 'calda' una temperatura superiore ai 20°C. A seconda dall'interesse, le temperature possono essere colte a distanza temporale varia, per ottenere valutazioni sul mese, sulla stagione, sull'anno o, infine, sulla storia del pianeta.

Anche quest'esempio sarà propagato lungo le diverse fasi della discussione di questa tesi e progressivamente esteso con i nuovi costrutti. Anche ad esso sarà dedicata una parte della progettazione ed implementazione.

Le località sono individui Dinamici, con una Linea Universo ed aventi un Evento per ogni temperatura colta.

Il *KB* contiene i seguenti concetti e ruoli.

$KB \supseteq \{Locality \sqsubseteq Dynamic;\ hasTemperature : Event \rightarrow Temperature;\ fun(hasTemperature);\ Temperature \sqsubseteq [-40.0, 60.0] \sqcap Rational;\ Warm = Temperature \sqcap (\geq 20.0);\ \geq 20.0 \equiv Trap(20, 20, 60, 60, -40, 60)\}$ (Figura 3-24, Figura 3-25). Inoltre, introduciamo le seguenti definizioni.

$WarmLocality = Locality \sqcap \exists hasTemperature.Warm$ è il concetto insieme degli individui evento (istanze nel tempo di località) aventi temperatura 'calda'.

$SometimeWarmLocality = ULine \sqcap \Diamond WarmLocality$ è il concetto insieme delle località (linee universo) che hanno avuto almeno una volta temperatura 'calda'.

$IncreasingWarmLocality = \nearrow(WarmLocality)$ è il concetto insieme delle località (linee universo) che si sono riscaldate e non si sono mai 'raffreddate' sensibilmente.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

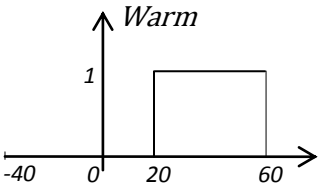

Figura 3-24 *Warm* crisp

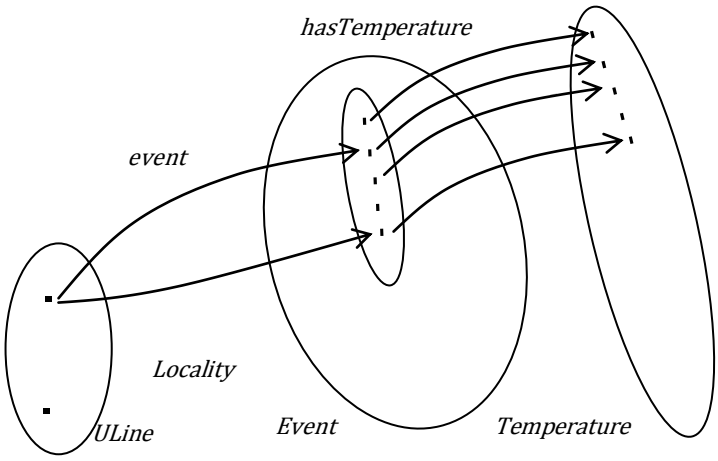

Figura 3-25 *Locality* e *Temperature*

ConstantWarmLocality = *SometimeWarmLocality* ⊓ ⇌(*WarmLocality*) è il concetto insieme delle località che sono state sempre 'calde'.

3.2.2 Modello temporale crisp $\mathcal{SHOIQT}$ (D)

Per ottenere una maggiore espressività del linguaggio, aggiungiamo $\mathcal{Q}$, le *qualified number restriction*. Mentre con $\mathcal{N}$ sono permesse espressioni come $\geq \mathrm{n} R$ e $\leq \mathrm{n} R$, dove R è un ruolo e n un intero, con $\mathcal{Q}$ sono permesse anche espressioni come $\geq \mathrm{n} R.C$ e $\leq \mathrm{n} R.C$, dove C è un concetto.

Mentre gli altri esempi non fanno uso di restrizioni in numero, l'esempio 1 si può arricchire nel seguente modo.

Esempio 1. Negozio di moda.

Nel nostro negozio di abbigliamento, si vuole riconoscere un premio ai clienti il cui numero d'acquisti di *prodotti di fascia alta* sia in crescita, in particolare ai clienti che da quando hanno compiuto più di 3 acquisti di *prodotti di fascia alta*, non sono mai scesi sotto questo numero. Si considera *prodotto di fascia alta* un prodotto il cui prezzo sia $\geq 200€$.

KB ⊇ { *buys* : *Event* → *Product* , *Customer* ≡ *Person* ⊓ ◇∃*buys*.⊤; *price* : *Product* → *PositiveRational*; *fun*(*price*); *HighPrice* = *PositiveRational* ⊓ (≥200); ≥200 ≡ *Trap*(200, 200, max, max, 0, max); *ExpensiveProduct* ≡ *Product* ⊓ ∃*price*.*HighPrice*} (Figura 3-26).

GoodCustomer = *Customer* ⊓ ≥3 *buys.ExpensiveProduct* è il concetto insieme degli individui evento (istanze nel tempo di clienti) che hanno compiuto più di 3 acquisti *di fascia alta*.

SometimeGoodCustomer = *ULine* ⊓ ◇*GoodCustomer* è il concetto insieme dei clienti (linee universo) che hanno compiuto almeno una volta più di 3 acquisti *di fascia alta*.

IncreasingGoodCustomer = ↗(*GoodCustomer*) è il concetto insieme dei clienti (linee universo) che hanno raggiunto la soglia dei tre acquisti *di fascia alta* e da allora non l'hanno mai lasciata.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

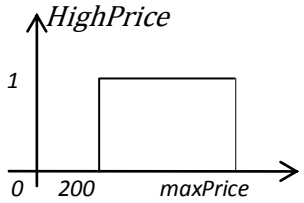

Figura 3-26 *HighPrice*, crisp

AwardedCustomer = *IncreasingGoodCustomer* è il concetto insieme dei clienti premiati.

3.2.3 Modello temporale fuzzy $f\_\mathcal{SHOINT}$ (D)

L'estensione fuzzy dei concetti espressi in logica $\mathcal{SHOINT}$ (D) crisp è ora abbastanza diretta, grazie alle definizioni che ho espresso precedentemente con 'un occhio al fuzzy'. Ho cioè espresso le mie definizioni in modo che fossero valide con concetti crisp, ma altrettanto valide con concetti fuzzy.

Abbiamo già detto come, nelle logiche fuzzy, i concetti contengano gli individui e i ruoli contengano le coppie di individui con un 'grado di appartenenza' o 'membership degree', appartenente all'intervallo unitario [0, 1]. In particolare, i concetti e i ruoli concreti hanno una propria membership-function esplicita, ad esempio *Cold* o *Young*, ma in generale essa non è necessaria, vedasi, ad esempio, *Blonde*, *Nice* o *friendOf*. Nella discussione dei capitoli precedenti, in logica crisp, si sono trattati implicitamente i concetti ed i ruoli come caso particolare dei corrispondenti fuzzy, restringendo per loro l'insieme dei membership degree a {0, 1}.

Tutta l'*Ontologia Superiore* temporale è costituita da concetti e ruoli crisp. I concetti *Static*, *Dynamic*, *ULine*, *Event*, *Time*; i ruoli *event*, *uline*, *time*, *now* sono tutti crisp e quindi nulla è da modificare. L'ordinamento di precedenza indotto in *Event* resta anch'esso inalterato.

I concetti, sia statici che dinamici, sono fuzzy: $C^{\mathcal{I}}(x) \in [0, 1]$ per ogni C e per ogni x.

La semantica dei concetti dinamici, ottenuti per azione degli operatori temporali *Until*, *Since*, *Sometime*, *Always*, resta immutata. Gli esempi possono essere estesi ai concetti dinamici fuzzy di 'Arricchito': *Enriched* = *Poor* $\mathcal{U}$ *Rich*, 'Dimagrito': *Slimmed* = *Fat* $\mathcal{U}$ *Slim*, 'Riscaldato': *WarmedUp* = *Cold* $\mathcal{U}$ *Warm* ecc., con *Until*, e 'FeliceDaQuandoSposato': *Happy* $\mathcal{S}$ *Married*, 'ArricchitoDa QuandoPromosso': *Enriched* $\mathcal{S}$ *Promoted*, 'FeliceDaQuandoDimagrito': *Happy* $\mathcal{S}$ *Slimmed*, con *Since*. Qui *Poor*, *Rich*, *Fat* ecc. sono tutti concetti fuzzy, con membership function esplicita o non.

In presenza di concetti il cui grado d'appartenenza può variare nel tempo, anche sensibilmente, eventualmente senza toccare mai gli estremi 0 e 1, si può tradurre nel linguaggio naturale la semantica del *Sometime* $\Diamond C^{\mathcal{I}}(x)$ come: 'x è stato C al più fino a questo grado'; quella dell'*Always* $\Box C^{\mathcal{I}}(x)$ come: 'x è stato sempre C o, perlomeno, non è mai sceso sotto a questo grado'. Analogamente per gli operatori *Until* e *Since,* che si possono comporre con *Sometime* ed *Always* (Figura 3-27).

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

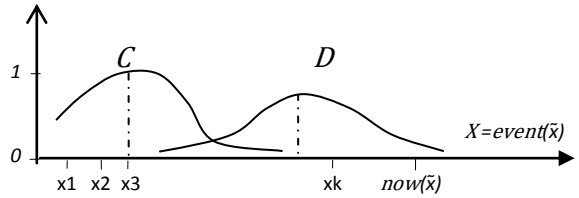

Figura 3-27 l'operatore *Until* : $\mathcal{CUD}$, fuzzy

I ruoli *firstTime* e *lastTime*, *nextTime* e *prevTime* su *Time*; *fistEvent* e *LastEvent*, *next* e *prev* su *Event*, sono ruoli crisp su insiemi crisp. Di conseguenza, anche gli operatori *Next* $\oplus$ e *Prev* $\ominus$ sui concetti mantengono semantica inalterata. Con essi, restano valide le definizioni iterative incrementali degli altri operatori temporali, presenti nella Tabella 3-2 .

Se consideriamo però la sussunzione tra due concetti fuzzy $A \sqsubseteq B$, abbiamo visto che $(A \sqsubseteq B)^{\mathcal{I}}$ è espressa da un numero appartenente all'intervallo unitario $[0, 1]$, calcolato col metodo *GD* (2-16), che rappresenta la quota di inclusione dell'insieme A in B. Abbiamo visto che, con concetti crisp, $SubsNext(A)^{\mathcal{I}} = (A \sqsubseteq \oplus A)^{\mathcal{I}}$ è un indice della crescenza (non stretta) dei valori in A assunti dagli individui dell'universo e in particolare, se x è una *ULine*, $SubsNext(A)^{\mathcal{I}}(x) = (A \sqcap X \sqsubseteq \oplus A \sqcap X)^{\mathcal{I}}$ è un indice della crescenza (non stretta) dei valori in A assunti dagli individui evento di x.

Resta vero che, per ogni x : *ULine*,

- $(A \sqcap X)^{\mathcal{I}}$ è crescente monotòna (non strettamente) se e solo se $SubsNext(A)^{\mathcal{I}}(x) = (A \sqcap X \sqsubseteq \oplus A \sqcap X)^{\mathcal{I}} = 1$.
- $(A \sqcap X)^{\mathcal{I}}$ è decrescente monotòna (non strettamente) se e solo se $SubsPrev(A)^{\mathcal{I}}(x) = (A \sqcap X \sqsubseteq \ominus A \sqcap X)^{\mathcal{I}} = 1$.
- $SubsNext(A)^{\mathcal{I}}(x) = SubsPrev(A)^{\mathcal{I}}(x) = 1$ se e solo se $(A \sqcap X)^{\mathcal{I}}$ è costante;

In tali casi di monotonia, infatti, si verifica la totale inclusione tra $(A \sqcap X)^{\mathcal{I}}$ e $(\oplus A \sqcap X)^{\mathcal{I}}$ oppure tra $(A \sqcap X)^{\mathcal{I}}$ e $(\ominus A \sqcap X)^{\mathcal{I}}$. Notiamo due fatti.

Primo, non valgono più le equivalenze con le (3-75), (3-76) e (3-77). Infatti, l'inclusione può essere totale di $(A \sqcap X)^{\mathcal{I}}$ in $(\oplus A \sqcap X)^{\mathcal{I}}$ − e quindi $SubsNext(A)^{\mathcal{I}}(x) = 1$ −, pur non raggiungendo mai $(A \sqcap X)^{\mathcal{I}}$ il valore 1. $\neg(A\ \mathcal{U} \neg A)^{\mathcal{I}}(x)$ registra invece i valori massimi raggiunti di A e $\neg A$ (quindi il massimo e il minimo di A) e non vale 1 se almeno uno dei due estremi non è stato raggiunto. Sembra dunque appropriato descrivere la crescenza e la decrescenza tramite i due operatori *SubsNext* e *SubsPrev*. Analogo ragionamento per le altre espressioni (3-76) e (3-77) che esprimono crescenza o decrescenza per mezzo degli operatori *Until* o *Since*.

Seconda cosa da notare è che può essere conveniente *sfruttare* il fuzzy, cioè il fatto che la sussunzione può dare un valore $\in [0, 1]$, per descrivere andamenti in generale oscillatori, non necessariamente sempre monotòni, e per chiedersi se 'mediamente' siano crescenti o decrescenti. Accade che, in assenza di monotonia, non è più vero che $SubsNext(C)^{\mathcal{I}} = \neg SubsPrev(C)^{\mathcal{I}}$ ed entrambi i valori possono essere non nulli.

Le caratteristiche che si constatano sono esposte nella Tabella 3-3.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Tabella 3-3 caratteristiche degli operatori *SubsNext* e *SubsPrev*

C		$SubsNext(C)^{\mathcal{I}}$	$SubsPrev(C)^{\mathcal{I}}$
monotono	crescente	1	0
	decrescente	0	1
	costante	1	1
oscillante		< 1	< 1
	crescente in media	$>$	
	decrescente in media	$<$	
	costante in media	$=$	
	maggiore l'ampiezza delle oscillazioni	minore il valore	minore il valore

Gli operatori *SubsNext* e *SubsPrev* danno la quota di perfetta inclusione, che corrisponde alla quota di perfetta monotonia. Ogni oscillazione rende nullo il contributo del corrispondente intervallo di α-cuts. Dalla Figura 3-28 si vede che per entrambi *SubsNext* e *SubsPrev* i contributi dell'α-range [1.0, 0.8] sono non nulli, diretta conseguenza del fatto che l'insieme vuoto $\phi \subseteq \phi$.

Da queste considerazioni, se siamo interessati allo studio dell'andamento in media dei valori di *C* assunti dagli individui nel tempo e dell'ampiezza delle loro oscillazioni, conviene ridefinire la semantica degli operatori *Incr*, *Decr* e *Const* nel seguente modo.

Definizione 11'. *Operatori d'Andamento* (*Tendency Operators*). Dato il concetto *C*, gli operatori '↗' (o *'Incr'*), '↘' (o *'Decr'*) e '⇌' (o *'Const'*), generano i seguenti nuovi concetti (*Concetti d'Andamento*, o *Tendency Concepts*), con la corrispondente semantica, espressa per x : *ULine*.

Increasing: $IncrC \equiv Incr(C) \equiv \nearrow C$.

$$\nearrow C^{\mathcal{I}}(x) \equiv IncrC^{\mathcal{I}}(x) = \max\{ SubsNextC^{\mathcal{I}}(x) - SubsPrevC^{\mathcal{I}}(x), 0.0\};$$

Decreasing: $DecrC \equiv Decr(C) \equiv \searrow C$.

$$\searrow C^{\mathcal{I}}(x) \equiv DecrC^{\mathcal{I}}(x) = \max\{ SubsNextC^{\mathcal{I}}(x) - SubsPrevC^{\mathcal{I}}(x), 0.0\};$$

Constant: $ConstC \equiv Const(C) \equiv \rightleftharpoons C$.

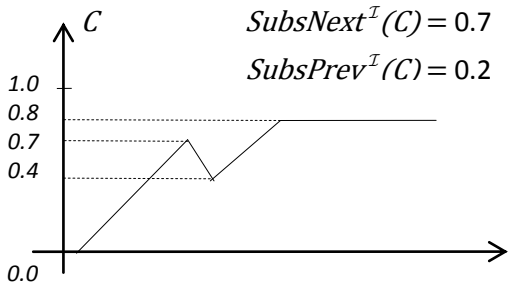

Figura 3-28 crescenza in media

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$$\rightleftharpoons C^{\mathcal{I}}(x) \equiv ConstC^{\mathcal{I}}(x) = \text{if } SubsNextC^{\mathcal{I}}(x) = SubsPrevC^{\mathcal{I}}(x) \text{ then } SubsNextC^{\mathcal{I}}(x) \text{ else } 0.0.$$

Questi concetti rappresentano dunque gli andamenti che, in media, sono rispettivamente crescenti, decrescenti e costanti e tengono conto sia dell'incremento o decremento che della monotonia dell'andamento.

$\nearrow C$ è quindi il concetto contenente quelle *ULine* i cui *Event* hanno valori in C *mediamente* crescenti, $\searrow C$ è il concetto contenente quelle *ULine* i cui *Event* hanno valori in C mediamente decrescenti e $\rightleftharpoons C$ è il concetto contenente quelle *ULine* i cui *Event* hanno valori in C mediamente costanti. Il grado d'appartenenza a $\nearrow C$ è tanto maggiore quanto maggiore è l'incremento e quanto minore è l'ampiezza delle oscillazioni. Analogamente per $\searrow C$, riguardo alla decrescenza, mentre per $\rightleftharpoons C$ si tiene conto solo dell'ampiezza delle oscillazioni. Gli andamenti 'medi' sono qui valutati in riferimento ad un'interpolazione lineare dei dati. Come esempi,

$\searrow$*Young* potrebbe essere il concetto degli individui esseri viventi sempre meno giovani (coincidente con l'intero insieme degli esseri viventi, con valore 1: invecchiamo tutti monotonicamente);

$\Diamond$*Warm* $\sqcap \rightleftharpoons$*Warm* potrebbe essere il concetto delle località costantemente calde;

$\nearrow$*Close* potrebbe essere il concetto degli individui che sono sempre più vicini;

$\nearrow$*Expensive* potrebbe essere il concetto dei prezzi sempre più cari;

$\nearrow$*Slim* potrebbe essere il concetto degli individui che sono sempre più magri; ecc., dove *Young*, *Warm*, *Close Expensive*, *Slim* sono concetti fuzzy.

Fornisco una definizione ulteriore per i Concetti d'Andamento e li chiamo *Monotonic Tendency Concepts*:

Definizione 11''. *Operatori d'Andamento Monotòno* (*Monotonic Tendency Operators*).
Dato il concetto C, gli operatori *'MonIncr'* e *'MonDecr'* generano i seguenti nuovi concetti (*Concetti d'Andamento Monotòno*, o *Monotonic Tendency Concepts*), con la corrispondente semantica, espressa per x : *ULine*.

MonotonicallyIncreasing: $MonIncrC \equiv MonIncr(C)$.

$$MonIncrC^{\mathcal{I}}(x) = \text{if } SubsNextC^{\mathcal{I}}(x) > SubsPrevC^{\mathcal{I}}(x) \text{ then } SubsNextC^{\mathcal{I}}(x) \text{ else } 0.0;$$

MonotonicallyDecreasing: $MonDecrC \equiv MonDecr(C)$.

$$MonDecrC^{\mathcal{I}}(x) = \text{if } SubsNextC^{\mathcal{I}}(x) < SubsPrevC^{\mathcal{I}}(x) \text{ then } SubsPrevC^{\mathcal{I}}(x) \text{ else } 0.0;$$

La differenza tra questi concetti e quelli della Definizione 11' è che, come dice il loro nome, la semantica dei *Concetti d'Andamento Monotòno* assegna un valore d'appartenenza che tiene conto solo della monotonicità dell'andamento, mentre la semantica dei *Concetti d'Andamento* della Definizione 11' assegna un valore d'appartenenza dipendente sia dalla monotonicità che dall'incremento (o decremento) raggiunto. Notiamo che per gli andamenti costanti non è necessaria la definizione di un nuovo concetto, poiché la semantica di $\rightleftharpoons C = ConstC$ tiene già conto unicamente della monotonicità.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Nel prosieguo della tesi prenderò in considerazione principalmente i *Concetti d'Andamento* della Definizione 11' piuttosto che quelli della Definizione 11''.

Esempio 1. Negozio di moda.

Per questo esempio, avevamo utilizzato nel KB il ruolo $buys : Event \rightarrow Product$, che è un ruolo crisp tra concetti crisp; inoltre la restrizione in numero ≥ 3 *buys* si potrà estendere solo dopo aver esteso il linguaggio con i quantificatori fuzzy. Per ora, quindi, lasciamo questo esempio inalterato.

Esempio 2. Clienti di banca virtuosi.

L'esempio dei clienti di banca può essere modificato nel seguente modo. Supponiamo si voglia definire un deposito 'rich' come un concetto fuzzy con membership function a trapezio, di valore certo (= 1) per un deposito ≥ 500 mila € , linearmente crescente per valori da 200 mila € a 500 mila € e nullo per valori ≤ 200 mila € (Figura 3-29).

La *KB* della banca contiene i seguenti concetti e ruoli:

$KB \supseteq \{Deposit \sqsubseteq Rational; hasDeposit : Event \rightarrow Deposit; fun(hasDeposit); Customer \equiv Person \sqcap \Diamond\exists hasDeposit.\top; RichDeposit \equiv Deposit \sqcap Rich; Rich \equiv MoreThan(\sim 500) \equiv Trap(200, 500, \max, \max, 0, \max)\}$. Inoltre, introduciamo le seguenti definizioni.

$RichDepCustomer = Customer \sqcap \exists hasDeposit.RichDeposit$ è il concetto insieme degli individui evento (istanze nel tempo di clienti) aventi deposito bancario 'rich'.

$SometimeRichDeposit = ULine \sqcap \Diamond RichDepCustomer$ è il concetto insieme dei clienti (linee universo) che hanno avuto almeno una volta un deposito 'rich'.

$VirtuousCustomer = \nearrow RichDepCustomer$ è il concetto insieme dei clienti (linee universo) la ricchezza del cui deposito è mediamente aumentata.

$ConstantRichCustomer = SometimeRichDeposit \sqcap \rightleftharpoons RichDepCustomer$ è il concetto insieme dei clienti linee universo il cui deposito è mediamente costantemente ricco.

Esempio 3. Agenti vicini al bersaglio.

In questo esempio si può *fuzzyficare* il concetto di 'vicino', definendolo come una distanza 'circa inferiore a 50 metri', che si annulli solo per distanze superiori a 60 metri (Figura 3-30):

Il *KB* contiene i seguenti concetti e ruoli.

$KB \supseteq \{Agent \sqsubseteq Dynamic ; hasDistance: Event \rightarrow Distanc ; Distance \sqsubseteq PositiveRational ;$

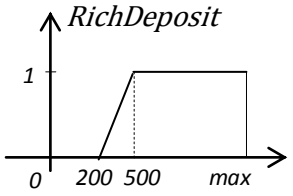

Figura 3-29 *RichDeposit*, fuzzy

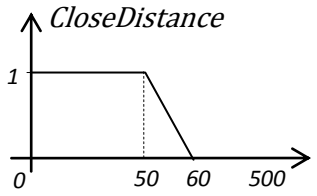

Figura 3-30 *CloseDistance,* fuzzy

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

CloseDistance ≡ *Distance* ⊓ *LessThan*(∼50); *LessThan*(∼50) ≡ *Trap* (0, 0, 50, 60, 0, 500)}. Ed, inoltre:

CloseAgent = *Agent* ⊓ ∃*hasDistance.CloseDistance* è il concetto insieme degli individui evento (istanze nel tempo di agenti) aventi distanza dal target 'vicina'.

SometimeCloseAgent = *ULine* ⊓ ◇*CloseAgent* è il concetto insieme degli agenti (linee universo) che sono stati almeno una volta a distanza dal target 'vicina'.

ApproachingAgent = ↗*CloseAgent* è il concetto insieme degli agenti (linee universo) che si sono mediamente avvicinati.

ConstantCloseAgent = *SometimeCloseAgent* ⊓ ⇌*CloseEvent* è il concetto insieme degli agenti mediamente costantemente vicini.

Esempio 4. Località calde.

Ridefiniamo il concetto di temperatura 'calda' in modo fuzzy, come quel valore che sia 'certo' (= 1) per temperature superiori a 20°C, sia 'falso' (= 0) per temperature ≤ 15°C e sia crescente linearmente per temperature tra 15°C e 20°C (Figura 3-31).

Il *KB* contiene i seguenti concetti e ruoli.

KB ⊇ {*Locality* ⊑ *Dynamic* ; *hasTemperature* : *Event* → *Temperature* ; *fun* (*hasTemperature*); *Temperature* ⊑ [-40.0, 60.0]⊓ *Rational* ; *Warm* ≡ *Temperature* ⊓ *MoreThan*(∼20); *MoreThan*(∼20) ≡ *Trap* (15, 20, 60, 60, -40, 60)} . Inoltre:

WarmLocality = *Locality* ⊓ ∃*hasTemperature.Warm* è il concetto insieme degli individui evento (istanze nel tempo di località) aventi temperatura 'calda'.

SometimeWarmLocality = *ULine* ⊓ ◇*WarmLocality* è il concetto insieme delle località (linee universo) che hanno avuto almeno una volta temperatura 'calda'.

IncreasingWarmLocality = ↗*WarmLocality* è il concetto insieme delle località (linee universo) che si sono mediamente riscaldate.

ConstantWarmLocality = *SometimeWarmLocality* ⊓ ⇌*WarmLocality* è il concetto insieme delle località che che sono state costantemente 'calde' in media.

3.2.4 Modello temporale fuzzy $f\_\mathcal{SHOIQ}\mathcal{T}$ (D)

Ammettendo nel linguaggio espressioni con restrizioni in numero qualificate $\geq n R.C$, $\leq n R.C$, dove sia i concetti che i ruoli sono fuzzy, con la semantica data dalla Tabella 2-3, l'espressività del linguaggio si arricchisce.

Esempio 1. Negozio di moda.

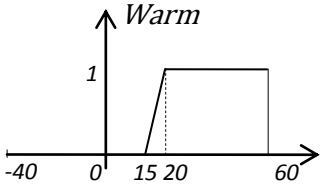

Figura 3-31 *Warm*, fuzzy

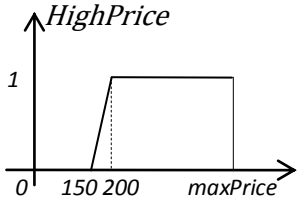

Figura 3-32 *HighPrice*, fuzzy

Ridefiniamo come '*HighPrice*' un prezzo, certamente se è superiore a 200€, con grado decrescente linearmente fino al valore 0 per 150€ (Figura 3-32). Conseguentemente, anche *ExpensiveProduct* diviene concetto fuzzy. I ruoli *buys* e *price* restano crisp.

$KB \supseteq \{ buys : Event \rightarrow Product , Customer \equiv Person \sqcap \Diamond\exists buys.\top ; price : Product \rightarrow PositiveRational ; fun(price) ; HighPrice \equiv PositiveRational \sqcap MoreThan(\sim 200) ; MoreThan(\sim 200) \equiv Trap(150, 200, \max, \max, 0, \max) ; ExpensiveProduct \equiv Product \sqcap \exists price.HighPrice \}$.

$GoodCustomer = Customer \sqcap \geq 3\ buys.ExpensiveProduct$ è il concetto insieme degli individui evento (istanze nel tempo di clienti) che hanno compiuto più di 3 acquisti di *'prodotti costosi'*.

$SometimeGoodCustomer = ULine \sqcap \Diamond GoodCustomer$ è il concetto insieme dei clienti (linee universo) che hanno compiuto almeno una volta più di 3 acquisti di *'prodotti costosi'*.

$IncreasingGoodCustomer = \nearrow GoodCustomer$ è il concetto insieme dei clienti (linee universo) che hanno aumentato il grado di 'buon cliente'.

$AwardedCustomer = IncreasingGoodCustomer$ è il concetto insieme dei clienti premiati.

3.2.5 Modello combinato: fuzzy quantificato e temporale $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Ricordiamo che in $\mathcal{ALCQ}^{+}$(D) (paragrafo 3.1) abbiamo introdotto i quantificatori come insiemi fuzzy, in particolare aventi una membership function lineare a tratti di forma trapezoidale. Grazie ad essi diventano lecite espressioni quantificate quali $QR.C$ e $Q\ A \sqsubseteq B$, con semantica data dal metodo *GD* (3-1, 3-2, 3-12, 3-13).

Estendiamo $f\_\mathcal{SHOIQT}$ (D) introducendo gli stessi quantificatori fuzzy ed otteniamo $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D), una classe di linguaggi di logica descrittiva ove è presente la dimensione temporale ed, in più, le qualified number restrictions sono estese con i più generali quantificatori fuzzy.

Determinate caratteristiche temporali del linguaggio, possono ora arricchirsi, consentendo espressioni quantificate. In più, dal versante dei quantificatori, essi possono essere estesi in modo naturale, a comprendere quantificatori con semantica temporale.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Abbiamo visto come i nuovi concetti *SubsNext(C)* e *SubsPrev(C)* siano correlati alla crescenza e decrescenza dei valori di un concetto C, assunti dall'insieme di eventi corrispondenti alla stessa linea universo, e abbiamo definito conseguentemente i concetti $IncrC \equiv \nearrow C$, $DecrC \equiv \searrow C$, $ConstC \equiv \rightleftharpoons C$. Questi concetti possono essere utilizzati in espressioni quantificate. Ad esempio,

(Most) Locality $\sqsubseteq \nearrow$ *WarmLocality* può corrispondere all'espressione linguistica 'la maggior parte delle località ha temperature mediamente sempre più calde';

(Some) Warm $\sqsubseteq \nearrow$ *WarmLocality* può corrispondere all'espressione linguistica 'alcune località calde hanno temperature mediamente sempre più calde';

(Most) Locality $\sqsubseteq \nearrow$ *TempLocality* può corrispondere all'espressione linguistica 'la maggior parte delle località ha temperature mediamente crescenti' ('riscaldamento globale'), avendo definito il concetto *TempLocality* $\equiv$ *Locality* $\sqcap \exists$ *hasTemperature.Temp* ed il concetto *Temp* con una membership function lineare crescente nelle temperature, ovvero un *Trap* di forma triangolare.

Oppure, se gestiamo un negozio di prodotti dimagranti,

Slimmed = *Fat* $\mathcal{U}$ *Slim;*

(AroundHalf) Customer $\sqsubseteq$ *Slimmed* corrispondente a 'circa la metà dei nostri clienti è dimagrita' (non sappiamo poi che sia successo),

(AroundHalf) Customer $\sqsubseteq \nearrow$ *Slim* corrispondente a 'circa la metà dei nostri clienti è sempre più magra' (continua a dimagrire).

Consideriamo inoltre il concetto D dato dall'espressione quantificata $D = QR.C$, ove R è un ruolo $R : A \rightarrow B$. Se A contiene solo elementi di *Static*, non abbiamo nulla da aggiungere. Se invece A contiene elementi di *Event*, allora $D \sqsubseteq$ *Event* e la quantificazione è 'istantanea' cioè riguarda individui evento nell'esatto istante di tempo a loro associato. Eventi, anche corrispondenti alla stessa Linea Universo, entrano in D indipendentemente, senza alcuna considerazione temporale.

Ad esempio, riallacciandoci all'esempio 1, se

$KB \supseteq \{$*buys* : *Customer* $\sqcap$ *Event* $\rightarrow$ *Product*, *ExpensiveProduct* = *Product* $\sqcap \exists$ *price.HighPrice*, *RichCustomer* $\equiv$ *(Most)buys.ExpensiveProduct* $\}$, allora

(Most)buys.ExpensiveProduct $\sqsubseteq$ *Event* contiene i singoli individui evento la cui maggior parte degli acquisti sono prodotti costosi.

Può invece interessare sapere quali siano i clienti, intesi come Linea Universo, ovvero come unica entità che si realizza nel tempo, che '*solitamente'* acquistano prodotti costosi, dove *'solitamente'* ha il significato di *'la maggior parte del tempo'* o *'la maggior parte delle volte',* in una dimensione temporale. Viene naturale l'identità: *Usually* $\equiv$ $Most_T$. Siamo necessitati a definire i *Quantificatori temporali* Q_T.

3.2.5.1 *Quantificatori temporali* Q_T

Nell'esempio sopra esposto, il ruolo *buys* non è definito su *ULine*. Su *ULine* è sempre definito il ruolo *event* che porta la singola *ULine* ai suoi eventi, le sue realizzazioni nel tempo. Ha senso

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

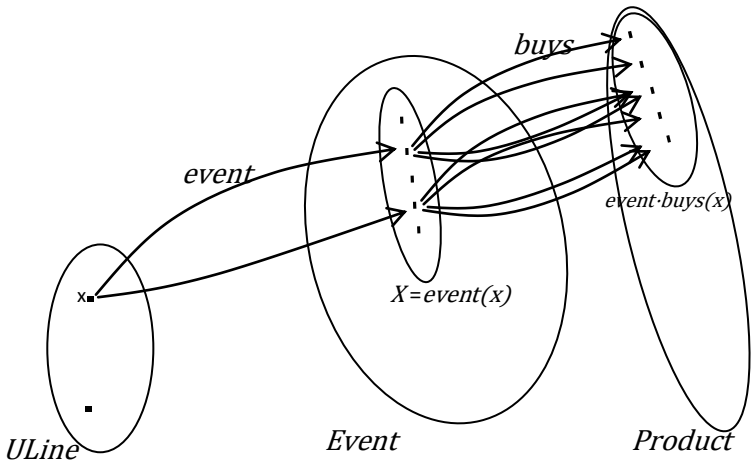

Figura 3-33 *event·buys(x)*

considerare la *composizione* di ruoli $event \cdot buys$ e, per x: *ULine*, la sua immagine $event \cdot buys$ (x), è l'unione di tutte le singole immagini di $buys(x_i)$ per ogni x_i appartenente a *event*(x) (Figura 3-33).

Si può allora definire il concetto

UsuallyRichCustomer ≡ *(Usually)buys.ExpensiveProduct* ≡ $(Most_T)$*buys. ExpensiveProduct* ≡ *(Most) event ·buys. ExpensiveProduct* ⊑ *ULine.*

Più in generale,

Definizione 14. Dato un quantificatore Q, si definisce il suo corrispondente *quantificatore temporale (temporal quantifier)* Q_T, che può essere partecipe di una espressione $Q_T R.C$, con R ruolo e C concetto, nel seguente modo:

$$Q_T \equiv (Q\, event \cdot) \qquad 3\text{-}80$$

da intendersi che la generica espressione $Q_T R.C$ si esplicita in $Q_T R.C \equiv Q\, event \cdot R.C$.

Si ha che $Q_T R.C \sqsubseteq ULine$, mentre $QR.C \sqsubseteq Event$.

I quantificatori temporali Q_T così definiti (da usarsi nelle espressioni $Q_T R.C$), inseriscono nelle espressioni logiche il ruolo *event*. Grazie a questa proprietà, possono essere utilizzati anche in una nuova forma espressiva, che ammettiamo in $\mathcal{SHOIQ}^{+}\mathcal{T}$ (D):

$$Q_T\, C = Q\, event.C \qquad 3\text{-}81$$

avente la conseguente semantica:

$Q_T\, C^{\mathcal{I}}(x) = (Q\, event.C)^{\mathcal{I}}(x) = GD_Q(C^{\mathcal{I}} / event^{\mathcal{I}}(x))$, se Q è un quantificatore relativo,

$Q_T C^{\mathcal{I}}(x) = (Q\, event.C)^{\mathcal{I}}(x) = GD_Q(event^{\mathcal{I}}(x) \cap C^{\mathcal{I}} / U^{\mathcal{I}})$, se Q è un quantificatore assoluto.

Il concetto $D = Q_T\, C$ è contenuto in *ULine* e corrisponde all'espressione linguistica 'insieme degli individui linee universo, per cui Q degli eventi sono C, cioè che Q del loro tempo o Q delle volte sono C'. Ad esempio,

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

(Usually)C = $Most_T C$ corrisponde a 'insieme degli individui che di solito sono *C*',

(MoreThanTwice)C = $MoreThan(2)_T\ C$ corrisponde a 'insieme degli individui che sono stati *C* almeno due volte',

$Between(\sim 2)and(\sim 5)_T C$ corrisponde a 'insieme degli individui che sono stati *C* tra le 2 e le 5 volte, circa'.

Esempio 4: località *solitamente* calde.

$WarmLocality \equiv Locality \sqcap \exists hasTemperature.Warm \sqsubseteq Event$

$UsuallyWarmLocality \equiv (Usually)WarmLocality \equiv (Most)_T\ WarmLocality = (Most)\ event.WarmLocality$

Si può anche introdurre una specializzazione Q_{now} del quantificatore temporale Q_T, ristretto all'evento avente *time* = *_now_*.

Definizione 15. Dato un quantificatore *Q*, si definisce il suo corrispondente *quantificatore attuale (now quantifier)* Q_{now}, che può essere partecipe di una espressione $Q_{now}R.C$, con *R* ruolo e *C* concetto, nel seguente modo:

$$Q_{now} \equiv (Q\ now \cdot) \qquad 3\text{-}82$$

da intendersi che l'espressione $Q_{now}R.C$ si esplicita in $Q_{now}R.C \equiv Q\ now \cdot R.C$.

Essa corrisponde all'espressione linguistica 'concetto insieme delle *ULine* per cui, al tempo attuale *time* = *_now_* , *Q* degli *R* sono *C*'.

Si ha che Q_{now} è una specializzazione di Q_T, nel senso che il ruolo *now* è una specializzazione del ruolo *event*.

Dimostro due teoremi a supporto della coerenza del sistema.

Teorema 1. Sussiste una relazione di equivalenza tra l'operatore *sometime* $\Diamond$ e il quantificatore temporale *Sometime*, quando *Sometime* sia definito come $Sometime \equiv Some_T$ e $Some \equiv \exists = \neg None = \neg Trap\ (0, 0, 0, 0)$; cioè, per ogni concetto *C*, per ogni x: *ULine* e per ogni interpretazione $\mathcal{I}$, si ha che $(Sometime)^{\mathcal{I}} C(\mathrm{x}) = \Diamond^{\mathcal{I}} C(\mathrm{x})$.

Dimostrazione.

Dalla definizione di *Trap*, si ha $None(\mathrm{y}) = Trap\ (0, 0, 0, 0)(\mathrm{y}) =$ if y=0 then 1 else 0.

Quindi $Some(\mathrm{y}) = 1 - None(\mathrm{y}) =$ if y=0 then 0 else 1. Dalla Definizione 3, ed equazione 2-14, abbiamo

$$GD_{Some}(G^{\mathcal{I}}/F^{\mathcal{I}}) = \sum_{\alpha_i \in \Lambda(G/F)} (\alpha_i - \alpha_{i+1}) Some\left(\frac{\left|(F \cap G)_{\alpha_i}\right|}{\left|F_{\alpha_i}\right|}\right) = \sum_{\substack{\alpha_i \in \Lambda(G/F) \\ \wedge \left(\frac{\left|(F \cap G)_{\alpha_i}\right|}{\left|F_{\alpha_i}\right|}\right) > 0}} (\alpha_i - \alpha_{i+1}) =$$

$\sum_{\substack{\alpha_i \in \Lambda(G/F) \\ \wedge\, F_{\alpha_i} \cap G_{\alpha_i} > 0}} (\alpha_i - \alpha_{i+1}) = \sup_{\mathrm{y}} (F^{\mathcal{I}} \cap G^{\mathcal{I}})(\mathrm{y})$. Allora,

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$(Some\ event.C)^{\mathcal{I}}(x) = GD_{Some}(C^{\mathcal{I}}/event^{\mathcal{I}}(x)) = \sup_y (event^{\mathcal{I}}(x) \cap C^{\mathcal{I}})(y) =$[8] $\sup_{y:X=event(x)} C^{\mathcal{I}}(y) = \Diamond^{\mathcal{I}} C(x)$.

q.e.d.

Teorema 2. Sussiste una relazione di equivalenza tra l'operatore *always* □ e il quantificatore temporale *Always*, quando *Always* sia definito come $Always \equiv All_T$ e $All \equiv \forall = Trap\ (1, 1, 1, 1)$; cioè, per ogni concetto C, per ogni x: *ULine* e per ogni interpretazione $\mathcal{I}$, si ha che $(Always)^{\mathcal{I}} C(x) = \Box^{\mathcal{I}} C(x)$.

Dimostrazione.

Si ha $All(y) = Trap(1, 1, 1, 1)(y) =$ if y=1 then 1 else 0. Quindi,

$$GD_{All}(G^{\mathcal{I}}/F^{\mathcal{I}}) = \sum_{\alpha_i \in \Lambda(G/F)} (\alpha_i - \alpha_{i+1}) All\left(\frac{|(F \cap G)_{\alpha_i}|}{|F_{\alpha_i}|}\right) = \sum_{\substack{\alpha_i \in \Lambda(G/F) \\ \wedge \left(\frac{|(F \cap G)_{\alpha_i}|}{|F_{\alpha_i}|}\right)=1}} (\alpha_i - \alpha_{i+1}) =$$

$$\sum_{\substack{\alpha_i \in \Lambda(G/F) \\ \wedge\ F_{\alpha_i} \subseteq G_{\alpha_i}}} (\alpha_i - \alpha_{i+1}).$$

Se F è crisp, si ha che $GD_{All}(G^{\mathcal{I}}/F^{\mathcal{I}}) = \inf_y (\neg F^{\mathcal{I}} \cup G^{\mathcal{I}})(y)$. Allora,

$(All\ event.C)^{\mathcal{I}}(x) = GD_{All}(C^{\mathcal{I}}/event^{\mathcal{I}}(x)) =$[9] $\inf_y (\neg event^{\mathcal{I}}(x) \cup C^{\mathcal{I}})(y) = \inf_{y:X=event(x)} C^{\mathcal{I}}(y) = \Box^{\mathcal{I}} C(x)$.

q.e.d.

3.2.5.2 *Alcune query dipendenti dal tempo*

Ricordiamo che, dichiarata la costante *_now_*, si era ricavato il concetto $Now = \exists time.\{\_now\_\}$, insieme degli individui aventi il tempo = *_now_* , ed il ruolo $now : ULine \sqcup Static \rightarrow Now$, con $now(x) = event(x) \sqcap Now$ per x:*Uline,* che invia dalla *ULine* all'evento corrispondente (unico) avente tempo $time = \_now\_$.

Sia chiaro che il valore effettivo della costante *_now_* è assegnato di volta in volta dalla particolare interpretazione, non dall'ontologia né, tanto meno, dall'*O*ntologia. Definita un'ontologia, si può scegliere un particolare t:*Time* e assumere che il valore di *_now_* sia $\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$. Da come abbiamo visto, la semantica di tutti i concetti dinamici ha la costante *_now_* come punto di riferimento, che diviene l'origine dell'asse temporale e il discrimine tra 'passato' e 'futuro'. L'interpretazione può compiere una scelta sull'origine dell'asse temporale, compiendo un'opportuna traslazione e facendola coincidere con l'istante t prescelto. In seguito, ogni inferenza logica può essere compiuta.

[8] *event* è un ruolo crisp, quindi $X = event(x)$ è un concetto crisp.
[9] v. nota precedente.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

In particolare, si possono formulare query dipendenti da un particolare istante di tempo Qry(t), che si risolvono assegnando $\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$ e traducendo la Qry(t) in Qry(*_now_*). Esempi si trovano nella Tabella 3-4.

Qualunque sia l'origine dell'asse temporale, una query temporale sull'appartenenza di un dato evento ad un concetto, sia esso statico o temporale, non dipende da essa, dunque $\mathrm{Qry}(t) = \langle a_i : C \rangle(t)? \rightarrow \mathrm{Qry} = \langle a_i : C \rangle?$ si traduce nella usuale query d'*instance check*.

Invece, il valore d'appartenenza di una *ULine* ad un concetto dipende strettamente dalla costante *_now_* e $\mathrm{Qry}(t) = \langle a : C \rangle(t)? \rightarrow \_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; $\mathrm{Qry} = \langle a : C \rangle?$ si traduce nell'usuale query d'*instance check*, solo dopo l'assegnazione di *_now_*.

Il terzo tipo di query è un'*instance check* indiretta: senza conoscere l'individuo evento, si chiede qual è il valore assunto dall'evento della *ULine* al tempo t, nel concetto C. Essa si scrive $\mathrm{Qry}(t) = \langle a : \exists event.C \rangle(t)$. In tal caso, dopo aver posto $\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$, si traduce nella query indipendente da t: $\langle now(a) : C \rangle$.

Per le query di tipo *retrieval*, per qualsiasi tipo di concetto ricercato, la forma dipendente dal tempo si traduce nella fissazione della costante *_now_* e nella mera traduzione nella query time independent: $\mathrm{Qry}(t) = \langle C \rangle(t)? \rightarrow \_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; $\mathrm{Qry}(\_now\_) = \langle C \rangle?$. In particolare, se si ricercano solo gli eventi appartenenti a C nell'istante t, la query $\mathrm{Qry}(t) = \langle Event \sqcap C \rangle(t)?$ si traduce in →

Tabella 3-4 alcune query dipendenti dal tempo

Qry(t)	Qry(*_now_*)	semantica in ling. naturale
$\langle a : C \rangle^{\mathcal{I}}(t)?$	$\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; $\langle a : C \rangle^{\mathcal{I}}?$	'qual è il valore d'appartenenza di a (*ULine*) al concetto C al tempo t'?
$\langle a : \exists event.C \rangle^{\mathcal{I}}(t)?$	$\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; $\langle now(a) : C \rangle^{\mathcal{I}}?$	'qual è il valore d'appartenenza al concetto C dell'evento di a al tempo t?'
$\langle a_i : C \rangle^{\mathcal{I}}(t)?$	$\langle a_i : C \rangle^{\mathcal{I}}?$	'qual è il valore d'appartenenza del particolare evento a_i al concetto C al tempo t'?
$\langle C \rangle^{\mathcal{I}}(t)?$	$\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; $\langle C \rangle^{\mathcal{I}}?$	'quali individui appartengono al concetto C al tempo t?'
$\langle Event \sqcap C \rangle^{\mathcal{I}}(t)?$	$\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; $\langle C \sqcap Now \rangle^{\mathcal{I}}?$	'quali eventi appartengono al concetto C al tempo t?'
$\langle ULine \sqcap C \rangle^{\mathcal{I}}(t)?$	$\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; $\langle C \sqcap ULine \rangle^{\mathcal{I}}?$	'quali linee universo appartengono al concetto C al tempo t?'
$\langle A \sqsubseteq B \rangle^{\mathcal{I}}(t)?$	$\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; $\langle A \sqsubseteq B \rangle^{\mathcal{I}}?$	'qual è il grado di sussunzione tra i due concetti A e B al tempo t?'

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; Qry(*_now_*) = ⟨*Now* ⊓ *C*⟩?; mentre la ricerca delle sole linee universo appartenenti a *C* nell'istante t, la query Qry(t) = ⟨*ULine* ⊓ *C*⟩ (t)? si traduce in → $\_now\_^{\mathcal{I}} = t^{\mathcal{I}}$; Qry(*_now_*) = ⟨*ULine* ⊓ *C*⟩?.

3.2.5.3 *Query di sussunzione*

Con la dimensione temporale definita con questa *Upper Ontology*, abbiamo la possibilità di porre delle query particolari. La sussunzione

A ⊑ *B* coinvolge in generale tutti gli individui, *Static*, *Uline* e *Event*. Il grado di verità della sussunzione è il grado di verità dell'affermazione che per ogni x, $A(x) \leq B(x)$. Ricordiamo che ogni evento è caratterizzato da un preciso istante temporale. Dunque[10],

A ⊓ *Uline* ⊑ *B,* è una sussunzione che coinvolge solo le *ULine*, quando su esse sono definiti *A* e *B* (ad esempio, se *A* e *B* sono definiti con operatori temporali). Invece,

A ⊓ *Event* ⊑ *B,* è una sussunzione che coinvolge gli *Event*: riguarda i valori istantanei e distribuiti nel tempo.

A ⊓ *Now* ⊑ *B,* è una sussunzione che coinvolge i soli *Event* attuali, aventi *time* = *_now_*.

A ⊓ *X* ⊑ *B,* è una sussunzione che coinvolge i soli *Event* relativi ad un'unica *ULine* (avendo dichiarato *X* = *event*(x), x:*ULine*). Ad esempio, con

InLove ⊑ *Happy ,* diciamo che 'ogni Innamorato è almeno altrettanto Felice'. Con

(Sometime)InLove ⊓ *ULine* ⊑ *(Sometime)Happy,* diciamo che 'ogni individuo che è stato Innamorato è stato anche almeno altrettanto Felice (non necessariamente nello stesso tempo)'. Invece, con

InLove ⊓ *Event* ⊑ *Happy,* diciamo che 'ogni individuo, quando è Innamorato, è anche (nello stesso tempo) almeno altrettanto Felice'; con

InLove ⊓ *Now* ⊑ *Happy,* diciamo che 'ora, al tempo attuale, tutti gli individui Innamorati sono anche (almeno altrettanto) Felici'; con

InLove ⊓ *Mary* ⊑ *Happy,* diciamo che 'mary è sempre stata almeno altrettanto Felice che Innamorata' (dove *Mary* = *event*(mary), mary:*ULine*); con

InLove ⊓ *Mary* ⊓ *Now* ⊑ *Happy,* diciamo che 'ora mary è almeno altrettanto Felice che Innamorata'.

Queste sussunzioni possono essere anche quantificate, es:

(Most) InLove ⊑ *Happy,* corrisponde all'affermazione che 'tra tutti gli individui Innamorati, la maggior parte è Felice'. Con

[10] Vale *A* ⊓ *C* ⊑ *C ;* quindi *A* ⊓ *C* ⊑ *B* equivale a *A* ⊓ *C* ⊑ *B* ⊓ *C*.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

(Most)((Sometime) InLove ⊓ *ULine* ⊑ *(Sometime)Happy),* diciamo che 'la maggior parte degli individui che sono stati Innamorati sono stati anche Felici (anche non contemporaneamente)'. Invece, con

(Most) InLove ⊓ *Event* ⊑ *Happy,* diciamo che 'la maggior parte degli individui Innamorati, in ogni tempo, è anche contemporaneamente Felice'; con

(Most) InLove ⊓ *Now* ⊑ *Happy,* diciamo che 'ora, al tempo attuale, la maggior parte degli individui Innamorati è anche Felice'; con

(Most) InLove ⊓ *Mary* ⊑ *Happy,* diciamo che 'mary è stata, per la maggior parte (solitamente), quando Innamorata anche Felice' (dove *Mary* = *event*(mary)). Invece

(Most) InLove ⊓ *Mary* ⊓ *Now* ⊑ *Happy,* non ha molto significato, essendo il concetto *Mary* ⊓ *Now* a cardinalità al più unitaria.

Abbiamo notato allora che:

(Q)A ⊑ *B* e *(Q)A* ⊓ *Event* ⊑ *B* sono quantificazioni globali (Figura 3-34);

(Q)A ⊓ *Now* ⊑ *B* è una quantificazione 'trasversale', sugli eventi corrispondenti alle diverse *ULine*, a tempo fissato (Figura 3-35);

(Q)A ⊓ *X* ⊑ *B* è una quantificazione 'temporale', sugli eventi di una fissata *ULine* (Figura 3-36).

Esempio 1. Negozio di moda.

Possiamo arricchire la *Knowledge Base* con alcuni nuovi concetti.

KB ⊇ {*buys*: *Event* → *Product* ; *Customer* ≡ *Person* ⊓ ◇∃*buys*.⊤; *price*: *Product* → *PositiveRational*; *fun*(*price*); *HighPrice* = *PositiveRational* ⊓ *MoreThan*(~200); *MoreThan*(~200) = *Trap* (150, 200, max, max, 0, max); *ExpensiveProduct* = *Product* ⊓ ∃*price*.*HighPrice; Sometime* ≡ $Some_{\mathcal{T}}$ *; Some* ≡ ¬*None* = ¬*Trap* (0, 0, 0, 0)}.

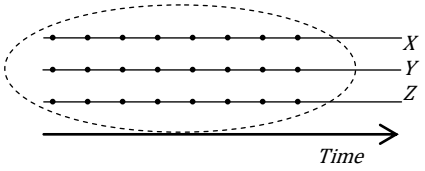

Figura 3-34 quantificazione globale

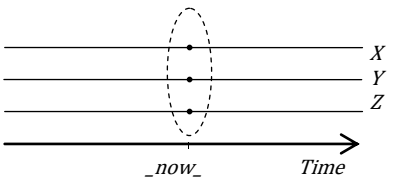

Figura 3-35 quantificazione trasversale

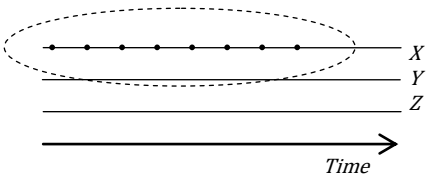

Figura 3-36 quantificazione temporale

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

RichCustomer = *Customer* ⊓ *(Most) buys.ExpensiveProduct* è il concetto insieme degli individui evento (istanze nel tempo di clienti) per cui la maggior parte degli acquisti sono *'prodotti costosi'*;

SometimeRichCustomer = *ULine* ⊓ *(Sometime)RichCustomer* è il concetto insieme dei clienti (linee universo) che hanno compiuto almeno una volta acquisti, per la maggior parte, di *'prodotti costosi'*;

UsuallyRichCustomer = *(Usually)RichCustomer* = *(Most$_T$)* (*Most*) *buys. ExpensiveProduct* è il concetto insieme dei clienti (linee universo) che solitamente compiono acquisti per la maggior parte di *'prodotti costosi'*.

Esempio 2. Clienti di banca virtuosi.

Possiamo inserire un altro concetto, la cui semantica è interessante: *UsuallyRichCustomer.*

La *KB* contiene, tra gli altri, i seguenti concetti:

Sometime ≡ *Some$_T$; Some* ≡ ¬*None* = ¬*Trap* (0, 0, 0, 0).

RichDepCustomer = *Customer* ⊓ ∃*hasDeposit.RichDeposit* è il concetto insieme degli individui evento (istanze nel tempo di clienti) aventi deposito bancario 'rich'.

SometimeRichDeposit = *ULine* ⊓ *(Sometime)RichDepCustomer* è il concetto insieme dei clienti (linee universo) che hanno avuto almeno una volta un deposito 'rich'.

VirtuousCustomer = ↗*RichDepCustomer* è il concetto insieme dei clienti (linee universo) la ricchezza del cui deposito è mediamente aumentata.

ConstantRichCustomer = *SometimeRichDeposit* ⊓ ⇌*RichDepCustomer* è il concetto insieme dei clienti linee universo il cui deposito è mediamente costantemente ricco.

UsuallyRichCustomer = *(Usually)RichDepCustomer* = *(Most$_T$)hasDeposit. RichDeposit* è il concetto insieme dei clienti linee universo che solitamente hanno un deposito bancario ricco.

Esempio 3. Agenti vicini al bersaglio.

Il *KB* contiene i seguenti concetti e ruoli.

KB ⊇ {*Agent* ⊑ *Dynamic* ; *hasDistance* : *Event* → *Distance* ; *Distance* ⊑ *PositiveRational* ; *CloseDistance* = *Distance* ⊓ LessThan(∼50) ; LessThan(∼50) = *Trap* (0, 0, 50, 60, 0, 500); *Sometime* ≡ *Some$_T$; Some* ≡ ¬*None* = ¬*Trap* (0, 0, 0, 0) } . Ed, inoltre:

CloseAgent = *Agent* ⊓ ∃*hasDistance.CloseDistance* è il concetto insieme degli individui evento (istanze nel tempo di agenti) aventi distanza dal target 'vicina'.

SometimeCloseAgent = *ULine* ⊓ *(Sometime)CloseAgent* è il concetto insieme degli agenti (linee universo) che sono stati almeno una volta a distanza dal target 'vicina'.

ApproachingAgent = ↗*CloseAgent* è il concetto insieme degli agenti (linee universo) che si sono mediamente avvicinati.

ConstantCloseAgent = *SometimeCloseAgent* ⊓ ⇌*CloseAgent* è il concetto insieme degli agenti mediamente costantemente vicini.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

UsuallyCloseAgent = *(Usually)CloseAgent* = $(Most_T)$ *hasDistance.CloseDistance* è il concetto insieme degli agenti solitamente vicini al target.

$isTargetFor : Target \rightarrow Agent \sqcap ULine$

TargetInDanger = *(Most)isTargetFor. UsuallyCloseAgent* è il concetto insieme degli obiettivi i cui agenti predatori sono solitamente vicini.

Esempio 4. Località calde.

Anche qui si possono inserire i quantificatori temporali nel KB e definire nuovi concetti.

$Sometime \equiv Some_T$; $Some \equiv \neg None = \neg Trap\,(0, 0, 0, 0)$;

$Usually \equiv Most_T = Most\ event \cdot$;

WarmLocality = *Locality* $\sqcap \exists$ *hasTemperature.Warm* è il concetto insieme degli individui evento (istanze nel tempo di località) aventi temperatura 'calda'.

SometimeWarmLocality = *ULine* $\sqcap$ *(Sometime)WarmLocality* è il concetto insieme delle località (linee universo) che hanno avuto almeno una volta temperatura 'calda'.

IncreasingWarmLocality = $\nearrow$*(WarmLocality)* è il concetto insieme delle località (linee universo) che si sono mediamente riscaldate.

ConstantWarmLocality = *SometimeWarmLocality* $\sqcap \rightleftharpoons$*(WarmLocality)* è il concetto insieme delle località che sono state costantemente 'calde' in media.

UsuallyWarmLocality = *(Usually)WarmLocality* è il concetto insieme delle località che che sono state solitamente 'calde'.

3.2.5.4 *Sintassi*

Riassumendo, le regole sintattiche ammesse in $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D), per i concetti, sono le seguenti.

Concetti: $C \rightarrow C_S \mid C_T$

Concetti Statici: $C_S \rightarrow \top \mid \bot \mid A_S \mid C_S \sqcap D_S \mid C_S \sqcup D_S \mid \neg C_S \mid QR.C_S \mid \{a\}$

La distinzione tra *Concetti Statici* e *Concetti Dinamici* è necessaria ai soli fini dell'ereditarietà dalle *ULine* agli *Event* corrispondenti.

I *Concetti Dinamici (Temporali)* sono costrutti contenenti un *Concetto Dinamico* o un *Quantificatore Temporale* oppure ottenuti tramite un *Operatore Temporale*.

Concetti Dinamici: $C_T \rightarrow tOp_1\, C \mid C\ tOp_2\, D \mid op_1\, C_T \mid C_T\, op_2\, D \mid C\, op_2\, D_T \mid tQ\ R.C \mid tQ\ C$

Operatori unari: $op_1 \rightarrow \neg \mid tOp_1$

verso $f\_\mathcal{SH}\mathcal{OIQ}^{+}\mathcal{T}$ (D)

Operatori Temporali unari: $tOp_1 \rightarrow \Diamond^{+} | \Diamond^{-} | \Box^{+} | \Box^{-} | \oplus | \ominus |$ *SubsNext* | *SubsPrev* | ↗ | ↘ | ⇌ | *MonIncr* | *MonDecr*

Operatori binari: $op_2 \rightarrow \sqcap | \sqcup | tOp_2$

Operatori Temporali binari: $tOp_2 \rightarrow \mathcal{U}^{+} | \mathcal{U}^{-} | \mathcal{S}^{+} | \mathcal{S}^{-}$

Quantificatori Temporali: $tQ \rightarrow Q_T | Q_{now}$.

4

Progettazione ed implementazione

In questo capitolo espongo la progettazione e l'implementazione di un'architettura software modulare, che rappresenti e gestisca una *Knowledge Base* ed esegua su di essa del semplice reasoning. Il nostro interesse è accentrato sulla valutazione di query contenenti i costrutti più originali definiti nella parte teorica.

Considereremo la valutazione di query di due tipi: $Q_T QR.C$ oppure $Q_T QA(op)B$, dove alcuni elementi possono o devono essere nulli. A, B, C sono concetti, R è un ruolo, Q è un quantificatore e Q_T è un quantificatore temporale. Infine, op è un operatore tra concetti, binario oppure unario, quando A è nullo.

In particolare ci interessano gli operatori temporali *sometime* $\Diamond$, *always* $\Box$, *Until* $\mathcal{U}$, *Since* $\mathcal{S}$, *Next* $\oplus$, *Prev* $\ominus$, *SubsNext*, *SubsPrev*, *Incr* $\nearrow$, *Decr* $\searrow$, *Const* $\rightleftharpoons$ e i concetti con questi ottenuti.

Gli algoritmi di valutazione seguono la semantica definita nella parte teorica. Per gli operatori per cui sono possibili anche algoritmi iterativi incrementali, sono stati implementati anch'essi.

4.1 Progettazione

Ho progettato un modulo denominato `reasoner`. Esso costituisce la business logic di ogni applicazione dimostrativa successivamente realizzata. Esso agisce in un dominio dato da qualsiasi sistema fisico in evoluzione temporale, cioè i cui dati che lo caratterizzano siano variabili col tempo.

Il modulo `reasoner` contiene in se' tutti gli algoritmi necessari per realizzare e gestire una *Knowledge Base*, contenente *TBox*, *ABox* e *Query*. Esso, naturalmente, non realizza un reasoner completo, con catene d'inferenze complesse – argomento di eventuali ricerche future – ma piuttosto si limita ai primi livelli d'inferenza e alla valutazione delle query di nostro interesse per l'estensione temporale sviluppata in questo lavoro.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Al modulo `reasoner` sono richieste le funzionalità della creazione di una *Knowledge Base*, della creazione, inserimento e rimozione in essa di Concetti, Concetti Concreti, Individui, Ruoli, Quantificatori e la creazione ed esecuzione di Query. Sono dunque necessarie le funzionalità corrispondenti alle operazioni logiche tra Concetti e Ruoli, previste dal linguaggio logico. In aggiunta, decidiamo che la nostra *Knowledge Base* dovrà contenere anche una funzionalità atta alla persistenza della propria *TBox* in un formato XML, il linguaggio maggiormente usato nell'interscambio dei dati (Figura 4-1).

In via preliminare, osserviamo che la suddivisione dell'universo, determinata dalla *Upper Ontology*, induce un'ontologia dell'individuo, quando l'individuo è inteso come qualunque istanza, materiale o spirituale, concreta o astratta, dell'universo. Il concetto di Individuo *Individual* ammette una partizione con i sottoconcetti Individuo Statico *StaticInd* e Individuo Dinamico *DynamicInd*; quest'ultimo, a sua volta, ammette una partizione con i sottoconcetti Individuo Evento *EventInd* e Individuo ULine *ULineInd* (Figura 4-2).

$StaticInd \sqsubseteq Individual$

$DynamicInd \sqsubseteq Individual$

$EventInd \sqsubseteq DynamicInd$

$ULineInd \sqsubseteq DynamicInd$

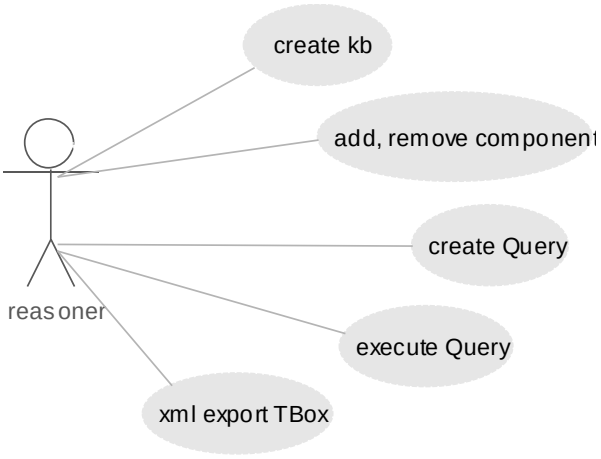

Figura 4-1 diagramma dei casi d'uso case per il modulo `reasoner`

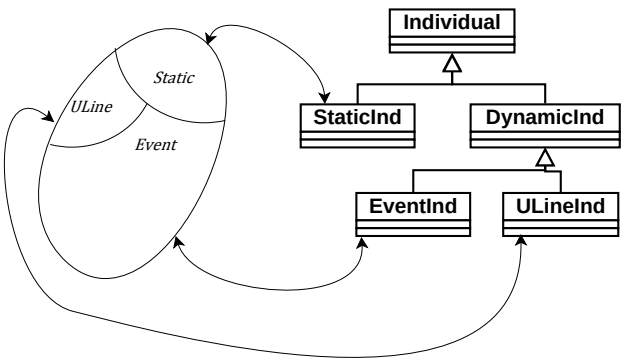

Figura 4-2 corrispondenza tra l'*Upper Ontology* e l'ontologia dell'Individuo

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

$Individual \equiv StaticInd \sqcup DynamicInd$

$StaticInd \sqcap DynamicInd \sqsubseteq \bot$

$DynamicInd \equiv EventInd \sqcup ULineInd$

$EventInd \sqcap ULineInd \sqsubseteq \bot$.

Adottando una visuale Object Oriented, definisco una classe `KnowledgeBase`, considerata come un contenitore di Concetti, Concetti Concreti, Individui, Ruoli, Quantificatori e di una Query. Inoltre essa contiene specificamente quei Concetti che ho denominato `ULineEvent` e che descriverò più avanti. In Figura 4-3 si vede un diagramma delle classi interessate, con aggregazioni e specializzazioni.

La classe `Concept` è vista come un contenitore di `Individual`. (Figura 4-3, Figura 4-4) Questa classe rappresenta i Concetti, che in generale sono fuzzy: dunque, ad ogni `Individual` deve essere associato un valore $\in [0, 1]$ che rappresenta il suo grado d'appartenenza al concetto.

La classe `Concept` è la classe più importante di tutto il modulo `reasoner`, poiché essa deve contenere anche le funzionalità che realizzano le operazioni sui concetti e che permettono la costruzione di nuovi concetti. Saranno presenti gli operatori statici *union* $\sqcup$, *intersection* $\sqcap$ e *negation* $\neg$, gli operatori temporali *sometime* $\Diamond$, *always* $\Box$, *until* $\mathcal{U}$, *since* $\mathcal{S}$ e di *sussunzione* $\sqsubseteq$ e gli operatori d'andamento *SubsNext*, *SubsPrev*, *Incr* $\nearrow$, *Decr* $\searrow$ e *Const* $\rightleftharpoons$, tutti che realizzano la semantica degli operatori come definita nel Capitolo 3.

Una particolare sottoclasse di `Concept` è la classe `ULineEvent`, che contiene solo quegli individui evento `EventInd` corrispondenti ad una stessa *ULine*, corrispondenti cioè allo stesso individuo `ULineInd`. Essa realizza ciò che nella parte teorica abbiamo spesso chiamato con le lettere *X*, *Y* maiuscole, cioè *X*: *ULineEvent* if ∃ x: *ULineInd* and *X* = *event*(x). (Figura 4-4).

La classe `Concept` implementa l'interfaccia `FuzzySet`, secondo la gerarchia illustrata dalla Figura 4-5.

Un'altra classe che implementa l'interfaccia `FuzzySet` è la classe `Trapezoid`. La classe `Trapezoid` rappresenta un trapezio in un dominio numerico, corrispondente alla funzione parametrica cui abbiamo dato il nome di *Trap*() (Definizione 4), da utilizzarsi come membership function per i concetti concreti e i quantificatori. Questi ultimi sono allora rappresentati dalle

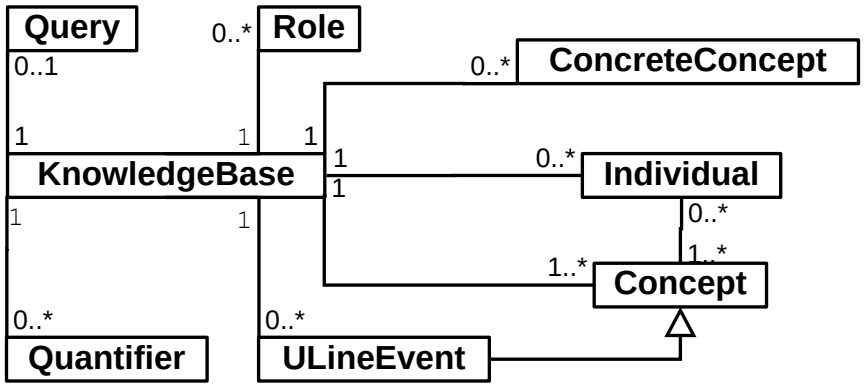

Figura 4-3 diagramma delle classi per il modulo `reasoner`

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

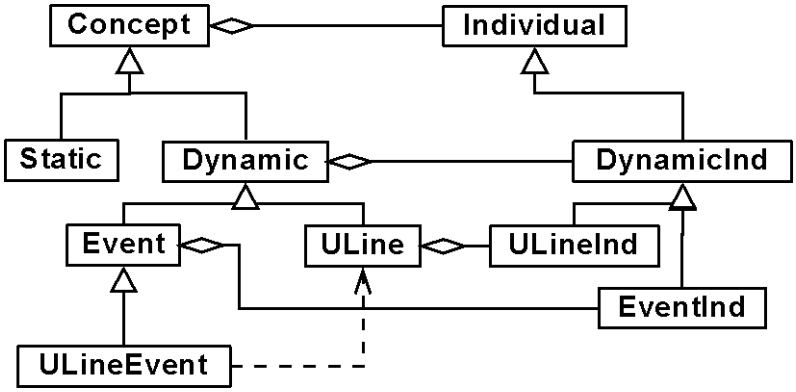

Figura 4-4 diagramma delle classi correlate a `Concept` ed `Individual`

classi `ConcreteConcept` e `Quantifier`.

Decidiamo di definire la classe `Role` come implementazione di `FuzzySet` anch'essa, poiché un ruolo è un insieme fuzzy che contiene coppie d'individui.

Infine, consideriamo un'interfaccia `Query` (Figura 4-6), che rappresenta la query generica e che è implementata dalle due classi `RoleQuery` e `OpQuery`. La classe `RoleQuery` rappresenta le query della forma $(Q_T)(Q)R.C$, con Q_T quantificatore temporale, Q quantificatore, R ruolo e C concetto. La classe `OpQuery` rappresenta invece le query della forma $(Q_T)(Q)C(\text{op})D$, con Q_T quantificatore temporale, Q quantificatore, C e D concetti e op operatore tra concetti.

4.2 Implementazione

Per l'implementazione è stato scelto il linguaggio Java, per la sua versatilità e la sua portabilità.

L'implementazione della classe `KnowledgeBase` include tutte le funzionalità per gli aggiornamenti degli insiemi di elementi che la compongono, quali l'aggiunta ed eliminazione di elementi, oltre ai vari metodi `getter` e `setter`. In aggiunta, `KnowledgeBase` conterrà anche, come detto, una funzionalità atta alla persistenza della propria *TBox* in formato XML. Una schematizzazione della struttura della classe `KnowledgeBase` è data dalla Figura 4-7. In essa risaltano i sei contenitori di `Concept`, `ConcreteConcept`, `Individual`, `Quantifier`,

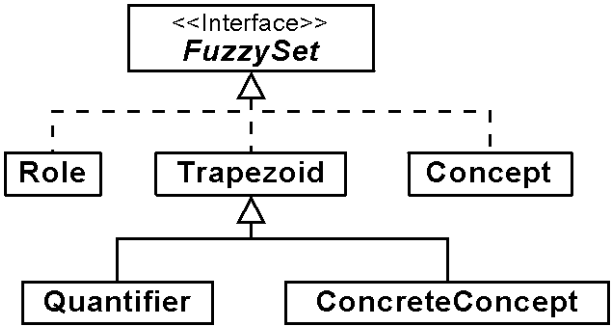

Figura 4-5 gerarchia dell'interfaccia `FuzzySet`

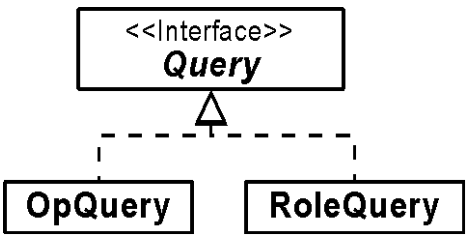

Figura 4-6 gerarchia dell'interfaccia `Query`

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

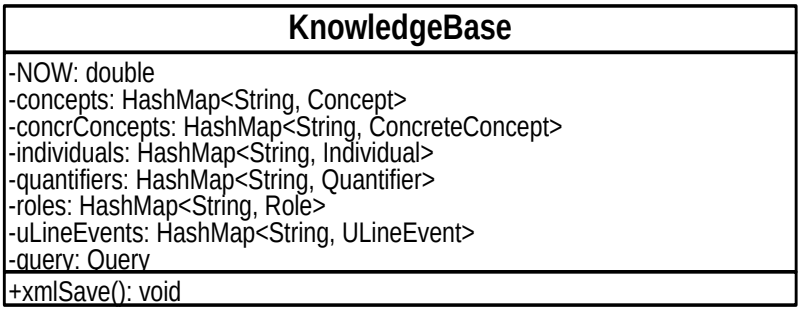

Figura 4-7 il diagramma della classe `KnowledgeBase`

`Role` e `ULineEvent`, realizzati tramite `HashMap`, che associano ad ogni elemento il suo nome univoco in formato `String`. È presente una costante denominata `NOW`, corrispondente alla costante del linguaggio logico *_now_*. Riguardo ai metodi, sono stati omessi dalla figura tutti i metodi d'aggiornamento ed i `getter` e `setter` – sia pur esistenti e necessari, essendo tutte le strutture private –, perché poco interessanti. È stato riportato invece il metodo `xmlSave()` che, appunto, salva la TBox della KB in formato XML. Della persistenza XML parlerò più avanti.

Anche nei diagrammi di classe che seguiranno, saranno omessi quei metodi `getter` e `setter` e i metodi d'aggiornamento che appaiono ovvii, per accedere agli attributi privati, al fine di non appesantire e mantenere invece l'attenzione focalizzata sui metodi più significativi.

La struttura della classe `Concept` si trova schematizzata nella Figura 4-8. Tre `Map<Individual, Double>` contengono le tre distinte classi di `Individual`. Il valore `Double` associato ad ogni `Individual` è il suo grado d'appartenenza al `Concept` stesso. L'attributo `kb` è un riferimento al `KnowledgeBase` d'appartenenza.

Sono presenti i metodi corrispondenti agli operatori statici e agli operatori temporali sui concetti, e che realizzano la semantica di questi operatori come definita nel Capitolo 3.

I metodi `union(Concept)`, `intersection(Concept)` e `negation()` restituiscono un nuovo `Concept` e corrispondono, rispettivamente, alle operazioni *union* ⊔, *intersection* ⊓ e *negation* ¬ .

I metodi `sometime(Individual)` ed `always(Individual)` restituiscono un valore `double`, il grado di verità delle espressioni, rispettivamente $\Diamond C$ (i) e $\Box C$ (i), dove C è il `Concept` attuale ed i l'`Individual`.

I metodi `getSometimeMap()` e `getAlwaysMap()` restituiscono i due attributi di tipo `Map`, `sometimeMap` e `alwaysMap`, che vengono usati come memoria delle ultime valutazioni, in ordine temporale, per i costrutti $\Diamond$, $\Box$, $\mathcal{U}$, $\mathcal{S}$ con algoritmo iterativo incrementale in accordo con la Tabella 3-2.

I metodi `until(Concept, Individual)`, `since(Concept, Individual)`, corrispondono rispettivamente alle espressioni $C\mathcal{U}D(\mathrm{i})$, $C\mathcal{S}D(\mathrm{i})$, con i soliti significati dei simboli; restituiscono un `double` esprimente il grado di verità dell'espressione. I metodi `until(Concept)` e `since(Concept)` restituiscono i nuovi `Concept`, corrispondenti, rispettivamente alle espressioni $C\,\mathcal{U}D$ e $C\,\mathcal{S}D$, essendo ancora C il `Concept` attuale e D il `Concept` argomento del metodo.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

I metodi `subsumed(Concept, boolean)` e `subsumed(Quantifier, Concept)` corrispondono alle espressioni logiche $C \sqsubseteq D$, $(Q)C \sqsubseteq D$, Nel metodo `subsumed(Concept, boolean)` il parametro `boolean` si pone `true` quando si desidera la sussunzione ordinaria crisp $\sqsubseteq$, `false` per la sussunzione fuzzy o quasi sussunzione $\tilde{\sqsubseteq}$.

Ricordo che a volte può essere conveniente l'uso della quasi-sussunzione $\tilde{\sqsubseteq}$ invece della sussunzione ordinaria $\sqsubseteq$ per la valutazione dei concetti di crescenza e decrescenza ↗, ↘, ⇌. Infatti, l'interpretazione di $\sqsubseteq$ fornisce solo la quota di effettiva monotonicità, cioè la percentuale di perfetta inclusione, come già visto. Nel caso, ad esempio, di presenza di un'oscillazione di ampiezza unitaria, pur seguita da regolare monotonia, la valutazione della $\sqsubseteq$ restituisce sempre 0. La quasi sussunzione, permette invece di recuperare e restituisce un valore che è tanto maggiore di 0, quanto più l'oscillazione tende ad essere isolata e ristretta nel tempo. Analogamente, quando l'andamento è crescente, ma con piccole oscillazioni continue, tali da occupare tutto il range dei valori di verità [0, 1], ancora solo con $\tilde{\sqsubseteq}$ si ottiene una valutazione positiva, mentre con $\sqsubseteq$ si ottiene 0. Ho lasciato comunque la possibilità di scelta del tipo di sussunzione, anche per poter fare un confronto in fase applicativa.

Il metodo `add(Individual, Double)` inserisce un nuovo `Individual` nel `Concept`

Concept
-events: Map<EventInd, Double> -uLineMap: Map<ULineInd, Double> -staticMap: Map<StaticInd, Double> -kb: KnowledgeBase -alwaysMap: Map<Individual, Double> -sometimeMap: Map<Individual, Double> -name: String #NOW: double
+union(Concept): Concept +intersection(Concept): Concept +negation(): Concept +sometime(Individual): double +getSometimeMap(): Map<Individual, Double> +always(Individual): double +getAlwaysMap(): Map<Individual, Double> +until(Concept,Individual): double +until(Concept,Indiviual,double): double +until(Concept): Concept +since(Concept,Individual): double +since(Concept): Concept +subsumed(Concept,boolean): double +subsumed(Quantifier,Concept): double +add(Individual,Double,): Concept +card(Double,String): Integer +getLambda(): TreeSet<Double> +getNextConcept(Individual): Concept +getLambda(): TreeSet<Double> +tempQ(Quantifier): Concept +createTendencyConcept(boolean) +subsNextPrev(boolean): ArrayList<Concept> +subsNext(boolean): Concept +subsPrev(boolean): Concept +getEvents(): Map<Individual, Double> +getULineMap(): Map<Individual, Double> +getKb()(): KnowledgeBase +getEvKeys(): Set<Individual> +containsKey(Individual): boolean +containsIndName(String): boolean +interpole(): Point +linearInterpolation2String(): String +toString(): String +print()

Figura 4-8 il diagramma della classe `Concept`

attuale o ne modifica il valore d'appartenenza, nel caso sia già presente.

I metodi `card()` e `getLambda()` sono ereditati dall'interfaccia `FuzzySet` e implementati.

Il metodo `getNextConcept(Individual)` fornisce il *Next* del `Concept` attuale: $\oplus C$ (i) per gli `Individual` precedenti quello dato in parametro. Nel caso che i sia *ULineInd*, il metodo fornisce $\oplus C$ completo per tutti i suoi *EventInd*.

Il metodo `TempQ(Quantifier)` costituisce un'implementazione dell'espressione $Q_T C$, dove Q_T è un `Quantifier` avente funzione di quantificatore temporale e C è il `Concept` corrente.

Il metodo `createTendencyConcepts()` chiama l'altro metodo `subsNextPrev()`, che, a sua volta, esegue più efficientemente le funzioni eseguite dai due metodi `subsNext()` e `subsPrev()`. `subsNext()` crea il nuovo `Concept` *SubsNext(C)*; `subsPrev()` crea il nuovo `Concept` *SubsPrev(C).* Il metodo `createTendencyConcepts()`, inoltre, per ogni individuo *ULine* `ULineInd`, esegue una valutazione comparativa e costruisce i *concetti d'andamento Incr(C)* (o $\nearrow C$), *Decr(C)* (o $\searrow C$) e *Const(C)* (o $\rightleftharpoons C$). In appendice si trova il codice dei due metodi `subsNext()` e `createTendencyConcepts()`.

Nella costruzione dei concetti *SubsNext(C)* e *SubsPrev(C)* e, conseguentemente, $\nearrow C$, $\searrow C$, $\rightleftharpoons C$, l'implementazione tiene conto, per efficienza, dell'equivalenza data dal Teorema 3 (che ho dimostrato in Appendice), che dice essere $(SubsPrevC)^{\mathcal{I}} \equiv (C \sqsubseteq \ominus C)^{\mathcal{I}} = (\oplus C \sqsubseteq C)^{\mathcal{I}}$.

Calcolato quindi $\oplus C$, viene utilizzato sia per il calcolo di $(SubsNextC)^{\mathcal{I}} \equiv (C \sqsubseteq \oplus C)^{\mathcal{I}}$ che di $(SubsPrevC)^{\mathcal{I}} = (\oplus C \sqsubseteq C)^{\mathcal{I}}$.

I metodi `getEvents()`, `getULineMap()` e `getKb()` restituiscono, rispettivamente, la `Map` di `EventInd`, la `Map` di `ULineInd` e la `KnowledgeBase` in comune a tutti i componenti.

Il metodo `getEvKeys()` restituisce una `TreeSet` contenente tutti gli individui evento `EventInd` appartenenti al `Concept` presente; `containsKey(Individual)` e `containsIndName(String)` sono test di presenza entro il `Concept`, rispettivamente dell'`Individual` a parametro o dell'`Individual` di cui si fornisce il nome.

In questa classe, come nelle altre, ogni metodo che riceve a parametro un `Individual`, distingue il caso di aver ricevuto un `EventInd` dal caso in cui a parametro sia un `ULineInd`, in accordo alle regole semantiche (3-41 / 3-50, 3-71, 3-72).

Il metodo `interpole()` è un metodo ausiliario, inserito nel `Concept` ai fini puramente di test, per una valutazione comparativa dei *concetti d'andamento*. Esso produce l'interpolazione lineare dei valori d'appartenenza al `Concept` presente, come una coppia <coefficiente angolare, intercetta>, istanza della classe `Point`. Questo metodo non è da considerarsi appartenente al *reasoning*. Ad esso è associato il metodo `linearInterpolation2String()`, che restituisce, in formato `String`, per ogni `ULineInd` il risultato dell'interpolazione lineare dei valori dei suoi `EventInd` nel `Concept` presente.

Infine, il metodo `toString()` offre in formato `String` una rappresentazione del Concept, con gli `Individual` ed i gradi d'appartenenza; `print()` stampa `toString()` nell'output di sistema.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Una particolare `Concept` è la classe `ULineEvent` (Figura 4-9), come ho accennato. Essa contiene tutti gli `EventInd` "fratelli", corrispondenti allo stesso `ULineInd`, e un riferimento all'`ULineInd` in comune. Una `TreeMap<Double, EventInd>` mantiene ordinato temporalmente l'insieme degli `EventInd`. Il valore `Double` nella `TreeMap` corrisponde al *time* dell'`EventInd` corrispondente. L'attributo `kb` è ancora il `KnowledgeBase` d'appartenenza.

La classe `ULineEvent` ha alcune funzionalità aggiuntive rispetto ad un `Concept` generale: può ricavare il primo e l'ultimo `EventInd`, tramite i metodi `getFirst()` e `getLast()`, corrispondenti ai ruoli *firstEvent* e *lastEvent* del linguaggio logico; e possiede i ruoli *next* e *previous* realizzati nei metodi `getNext()` e `getPrevious()`.

Inoltre, il metodo `get(double, Concept)` restituisce come `double` il grado d'appartenenza, al `Concept` in input, dell'individuo `EventInd` che appartiene alla `ULineEvent` attuale e che ha come *time* il `double` fornito in input; il metodo `getIndividual (double)` riceve il *time* e restituisce l'`Individual` corrispondente.

Il metodo `getMyULine()` restituisce il riferimento all'`ULineInd` cui tutti gli `EventInd` di questa classe corrispondono.

La classe astratta `Individual` è molto semplice (Figura 4-10): contiene gli attributi `name` di tipo `String` – il nome dell' `Individual` –, `time` di tipo `double` – il *time* dell'`Individual` – e `kb` – il `KnowledgeBase` –; i metodi `getter` e `setter` relativi, non rappresentati in figura, un metodo di comparazione lessicografica con un altro `Individual`, `compareTo(Individual)`, ed il metodo astratto `degree(Concept)`, che restituisce il grado d'appartenenza dell'`Individual` al `Concept` d'input.

La classe `Individual`, come abbiamo visto (Figura 4-2), ha la sua gerarchia di specializzazione in `StaticInd`, `DynamicInd`, `ULineInd` e `EventInd`. `DynamicInd` è ancora classe astratta, mentre `StaticInd`, `ULineInd` e `EventInd` sono concrete.

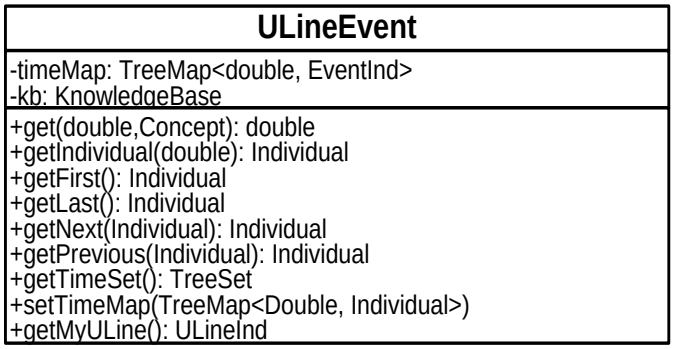

Figura 4-9 diagramma della classe `ULineEvent`

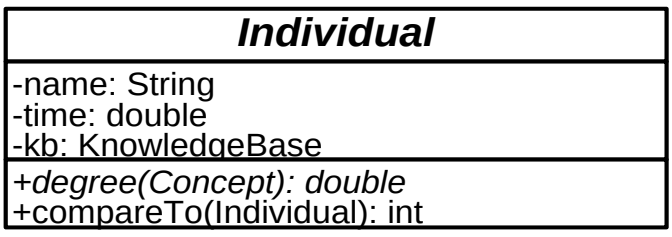

Figura 4-10 diagramma della classe astratta `Individual`

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Quindi, `StaticInd`, `ULineInd` e `EventInd` implementano il metodo `degree(Concept)`, ognuno andando a cercare nel rispettivo `Map` del `Concept`: `staticMap`, `ULineMap` oppure `EventMap`.

L'interfaccia `FuzzySet` espone le signatures di due soli metodi (Figura 4-11):

`getLambda()` restituisce l'insieme Λ dei livelli – α del `FuzzySet`;

`card(Double, String)` restituisce la cardinalità del `FuzzySet` al livello – α dato dal parametro `Double`; il parametro `String` indica la modalità di cardinalità desiderata, per eseguire una selezione sugli `Individual`.

La classe `Trapezoid`, come abbiamo detto, implementa la funzione *Trap*() (Definizione 4). Il suo diagramma di classe è dato dalla Figura 4-12. Gli attributi `double a`, `b`, `c`, `d` rappresentano i quattro vertici del trapezoide; `maxval` e `minval` sono i valori massimo e minimo del suo dominio; `name` e `meaning` sono il nome ed il significato espresso in linguaggio naturale nella forma `Around(n)`, `Between(n)and(m)`, `LessThan(n)`, `MoreThan(n)`, oltre ad alcuni trapezoid particolari.

Tra i metodi, risaltano le quattro operazioni `sum()`, `diff()`, `prod()` e `ratio()` eseguite tra il `Trapezoid` attuale ed un altro `Trapezoid` dato in input. Queste quattro operazioni, intese come operazioni tra fuzzy number, sono state implementate in modo da restituire come risultato sempre una funzione lineare a tratti di tipo `Trapezoid`. L'operazione `ratio()` è eseguibile solo quando il secondo `Trapezoid` (il divisore) non contiene lo zero.

Il metodo `fixVertices()` sistema i valori dei vertici affinché rispettino i vincoli dati da `minval` e `maxval`.

Importante è il metodo `qvalue(Double)`: esso esegue la valutazione del grado d'appartenenza al `Trapezoid`, inteso come membership function di un Fuzzy Set, del punto del dominio di coordinata data dal `Double` d'input.

La classe `Quantifier` è sottoclasse di `Trapezoid`. Essa rappresenta i quantificatori del linguaggio logico, implementati come trapezoidi che possono ammettere solo valori non negativi (Figura 4-13). Attributi significativi sono `kb`, il `KnowledgeBase` d'appartenenza e il `boolean relative` che indica se il `Quantifier` attuale è relativo o assoluto. Le operazioni tra `FuzzySet` sono ora specializzate tra `Quantifier` e restituiscono un `Quantifier`: somma e

Figura 4-11 l'interfaccia `FuzzySet`

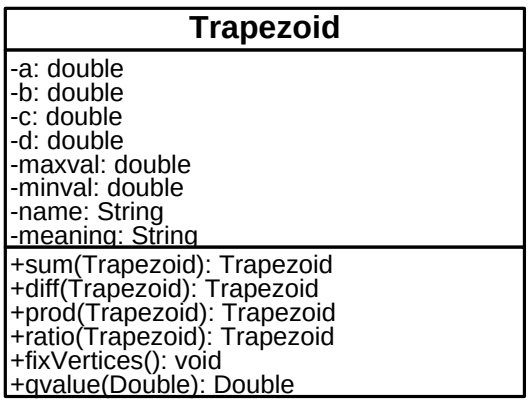

Figura 4-12 la classe `Trapezoid`

differenza sono possibili solo tra `Quantifier` assoluti e danno un assoluto; il prodotto tra due assoluti dà un assoluto, tra due relativi dà un relativo, tra un relativo e un assoluto dà un assoluto; il rapporto di un assoluto con un relativo dà un assoluto, di un relativo con qualsivoglia dà un relativo, di un assoluto con un assoluto dipende dai valori e dalla semantica che gli si vuole attribuire.

Il metodo `fixVertices()` specializza ai requisiti dei `Quantifier` il metodo omonimo della superclasse.

Il metodo `toString()` restituisce una rappresentazione in formato `String` del `Quantifier` attuale.

Il metodo `eval(Role, Concept)` esegue una valutazione dell'espressione logica $(Q)R.C$, implementando l'espressione GD (3-1, 3-2).

La classe `ConcreteConcept` (Figura 4-14) rappresenta i concetti concreti a membership function esplicita, di tipo trapezoidale. L'attributo `relative` dei `Quantifier` è assente; le operazioni di somma, differenza, prodotto e rapporto sono essenzialmente quelle di `Trapezoid`. È invece presente un ruolo `toConcept()`, che restituisce il `Concept` ad esso associato nella `KnowledgeBase` (quello con lo stesso nome). Notiamo che `ConcreteConcept` non contiene `Individual`, come invece fa `Concept`.

La classe `Role` rappresenta un ruolo logico o, meglio, la proiezione del ruolo su un individuo:

`Role` $\leftrightarrow R_x = \{y \mid (x, y) : R\}$, così da poter essere anch'essa considerata un fuzzy set contenitore d'individui, invece che di coppie d'individui (Figura 4-15). L'attributo `r` è la `Map<Individual, Double>` contenente l'insieme d'`Individual`, ognuno col suo grado d'appartenenza.

Tra i metodi, oltre a `getLambda()` e `card()`, implementazioni dei ruoli astratti ereditati dall'interfaccia, si nota `containsInd(Individual)`, che ricerca se un dato `Individual` è presente nel `Role` attuale; `totalLambda(Concept)`, che restituisce l'unione dei Λ del `Role` e del `Concept` a parametro. Il metodo `Intersection(Concept)` esegue l'intersezione tra il `Role` ed il `Concept` a parametro: questa operazione è possibile in quanto sono entrambi fuzzy set d'`Individual`; la semantica utilizzata per l'intersezione è quella del min, dell'intersezione standard. Questo metodo è richiamato nella valutazione che segue l'algoritmo *GD*, eseguita dal metodo `eval(Role, Concept)` della classe `Quantifier`.

L'interfaccia `Query` rappresenta la query generica (Figura 4-16). Essa presenta le signatures dei metodi seguenti: `evaluate()`, che esegue la valutazione della `Query` attuale e ne restituisce il grado di verità come un `double`; `getMeaning()`, che restituisce la semantica della `Query` attuale in formato `String`; `toString()`, che fornisce una rappresentazione `String` della

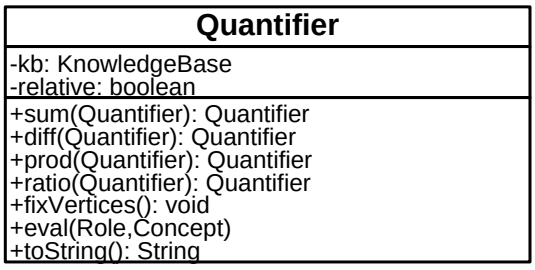

Figura 4-13 la classe `Quantifier`

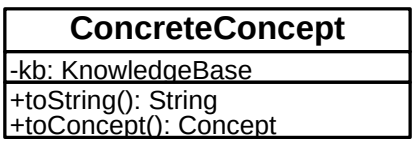

Figura 4-14 la classe `ConcreteConcept`

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

`Query` attuale.

L'interfaccia `Query` è implementata dalle classi `RoleQuery` e `OpQuery`.

La classe `RoleQuery` (Figura 4-17) rappresenta le query aventi forma $(Q_T)(Q)R.C$: i suoi attributi principali sono quindi `tempq` e `q`, di tipo `Quantifier`, `r` di tipo `Role`, `c` di tipo `Concept`, `tc` di tipo `ConcreteConcept` e `kb` di tipo `KnowledgeBase`. I metodi principali sono l'implementazione dei metodi dell'interfaccia `Query`, `evaluate()`, `getMeaning()` e `toString()`; inoltre è presente il metodo `trap2Concept()` che traduce un concetto trapezoidale `ConcreteConcept` nel corrispondente `Concept`.

La classe `OpQuery` (Figura 4-18) rappresenta le query aventi forma $(Q_T)(Q)A(\mathrm{op})B$; i suoi attributi principali sono `tempq` e `q` di tipo `Quantifier`, `c1` e `c2` di tipo `Concept`, `tc1` e `tc2` di tipo `ConcreteConcept`, `kb` di tipo `KnowledgeBase`, `operator` di tipo `String` e `lastValues`, una `TreeMap<String, Double>` adibita a contenere i risultati delle ultime valutazioni in ordine di tempo per l'esecuzione degli algoritmi iterativi incrementali. I metodi principali sono ancora l'implementazione dei metodi dell'interfaccia `Query`, `evaluate()`, `getMeaning()` e `toString()`; il metodo `trap2Concept()` che traduce un `ConcreteConcept` nel corrispondente `Concept`; il metodo `evaluateString(boolean, boolean)` che esegue una valutazione della `Query` ed emette un report in formato `String`. Dei suoi due parametri `boolean`, il primo indica se è richiesta una valutazione incrementale, il secondo se, in caso di sussunzione, la sussunzione debba essere valutata in modo fuzzy (corrispondente alla quasi – sussunzione $\tilde{\sqsubseteq}$) oppure crisp (la sussunzione ordinaria $\sqsubseteq$).

Infine, due parole riguardo alla persistenza della *TBox* della *Knowledge Base* in formato XML. Ho utilizzato la libreria Java JAXB, la quale permette il binding tra una struttura xml, definita da uno schema, ed alcune classi Java, preservando il livello concettuale dello schema stesso. Attraverso questa tecnica siamo in grado di generare in maniera automatica un mapping tra documenti xml e

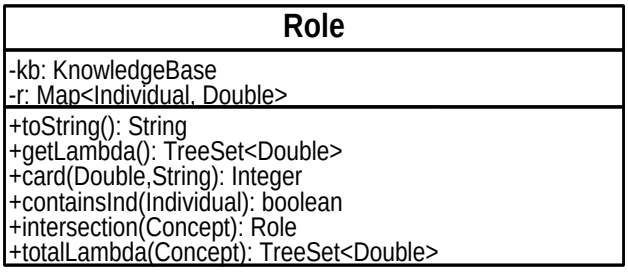

Figura 4-15 la classe `Role`

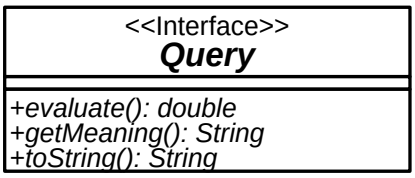

Figura 4-16 l'interfaccia `Query`

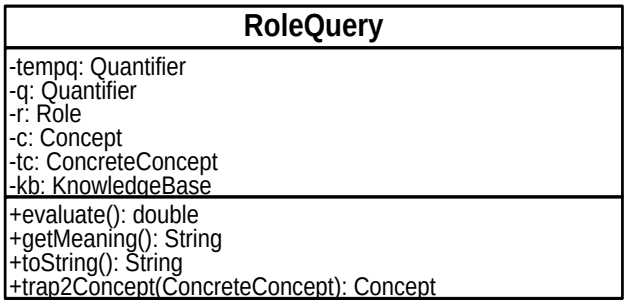

Figura 4-17 la classe `RoleQuery`

oggetti Java e viceversa. Il codice generato consente la gestione del documento xml a un livello superiore rispetto ai consueti SAX e DOM, pur mantenendo caratteristiche prestazionali di parsing simili a quelle di SAX e capacità di memorizzazione dei dati più efficienti rispetto a quelle di DOM.

Costruito uno schema xsd, si generano automaticamente delle classi Java che posso usare come ponte per il marshalling e unmarshalling del *TBox*. Teoricamente, potrei usare direttamente le classi cui fa riferimento la classe `KnowledgeBase`, se volessi la persistenza dell'intero kb e di tutte le sue funzionalità; in realtà ci basta la sua struttura dei concetti, ruoli, quantificatori e query. Lo schema TBox.xsd è visibile in appendice. Lo schema contiene un elemento di nome *TBox*, di tipo complesso TBoxType. Il tipo complesso TBoxType contiene a sua volta elementi di tipo complesso corrispondenti a liste di SimpleConcept, QuantifierType, RoleType e una QueryType. SimpleConcept e QuantifierType contengono a loro volta il tipo TrapeziumType. Ogni elemento e tipo ha i suoi attributi di tipo numerico o string. Da questo schema vengono generate le corrispondenti classi Java `TrapeziumType`, `SimpleConcept`, `QuantifierType`, `RoleType` e `QueryType`, oltre a `ObjectFactory`, che espone i metodi di factoring per la creazione degli oggetti. Una corrispondenza tra le classi è illustrata nel diagramma di Figura 4-19. Utilizzando queste classi, il marshalling della *TBox* è estremamente semplice. Il codice del metodo `xmlSave()` è in appendice.

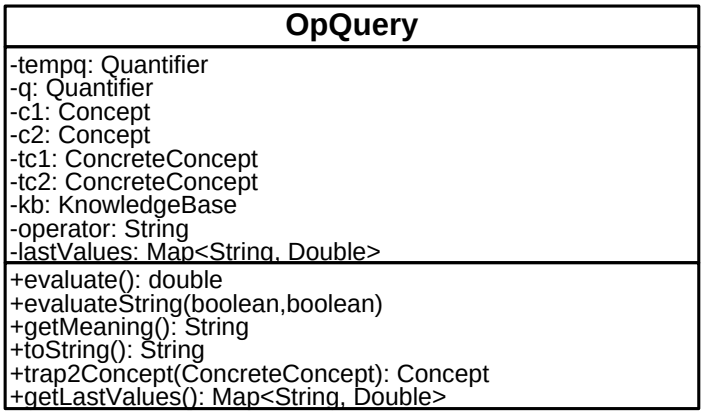

Figura 4-18 la classe `OpQuery`

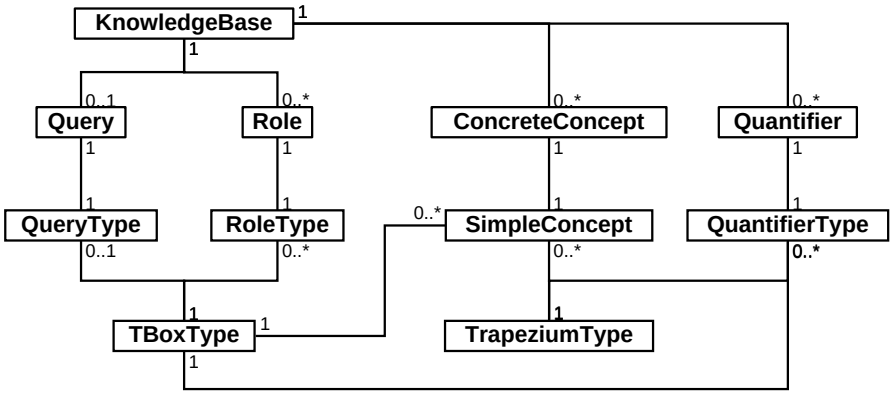

Figura 4-19 diagramma delle classi coinvolte nella traduzione xml

5

Applicazioni dimostrative

Questo capitolo contiene la descrizione della progettazione e dell'implementazione di alcune applicazioni dimostrative, che usano il modulo di reasoning oggetto del capitolo 4.

Nel paragrafo 5.1 un modulo di presentazione permette la scelta tra l'azione di editing e quella di monitoring.

In 5.2 un modulo di editing permette la creazione della *MBox* e *TBox* di una *Knowledge Base*, con la configurazione delle membership function dei concetti concreti e dei quantificatori, la creazione di una query e l'esportazione della *Knowledge Base* in formato XML. In 5.3 si presentano tre casi di monitoring, successivamente dettagliati.

In 5.4 si esamina il caso di un monitoring in tempo reale: nello specifico, si riprende e si implementa l'esempio già visto di un sistema dinamico di tipo preda – predatori. In questo caso, il *TBox* viene importato in formato XML prima di iniziare il monitoraggio, mentre l'*ABox* viene ripetutamente e periodicamente aggiornato con i nuovi dati sullo stato corrente importati da un file xml. Il file xml contenente i dati sullo stato corrente è infatti continuamente rinnovato da parte dell'emulatore del sistema dinamico. In seguito ad ogni importazione, il monitor riesegue anche la valutazione della query in esame ed aggiorna i risultati a video. Questo tipo di monitoraggio impone, per motivi di efficienza, l'esecuzione degli algoritmi iterativi incrementali, ove si valutino operatori che ammettano un tal tipo di semantica.

In 5.5 si progetta ed implementa un sistema di monitoring che importi l'*ABox* da un data base relazionale una volta per tutte o almeno con una frequenza così bassa da non necessitare particolari efficienze computazionali. Il monitor espone graficamente in un diagramma i dati importati nell'*ABox*, in modo che l'utente possa avere anche una percezione visiva degli andamenti. Si è utilizzato questo tipo di monitoring in 5.5.1, per la valutazione di query connesse al problema del riscaldamento del pianeta e in 5.5.2 per la valutazione di query a carattere sanitario.

5.1 Choosing

Un semplice modulo di presentazione funge da menu principale ed offre le possibilità di scelta tra:

- operazioni di editing del *Knowledge Base*
- operazioni di monitoring per valutare le query del *Knowledge Base*

Il progetto sviluppato segue il pattern MVC, *model view controller*. Il model contiene la business logic ed è costituito dal modulo denominato `reasoner` illustrato nel capitolo precedente.

All'utente si offrono le due alternative principali (Figura 5-1): gli si offre la scelta se aprire l'editor per creare una nuova *Knowledge Base* oppure se aprire uno dei monitor per eseguire il monitoraggio. Nella Figura 5-2 si vede l'interfaccia grafica del modulo di presentazione.

5.2 Editing

Nel caso che l'utente scelga l'editor, con esso può creare una query, naturalmente dopo aver creato le relative componenti, cioè i ruoli, concetti e quantificatori coinvolti, cioè l'*MTBox*, composto da *MBox* cui appartengono i quantificatori e *TBox*, cui appartengono i concetti e i ruoli. L'altra azione che può eseguire l'utente è l'esportazione dell'*MTBox* e della Query in formato xml (Figura 5-3).

L'implementazione offre un'interfaccia grafica all'utente, dalla quale si possono richiamare

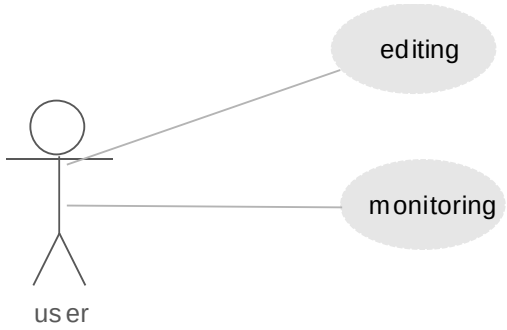

Figura 5-1 casi d'uso: scelta tra editor e monitor

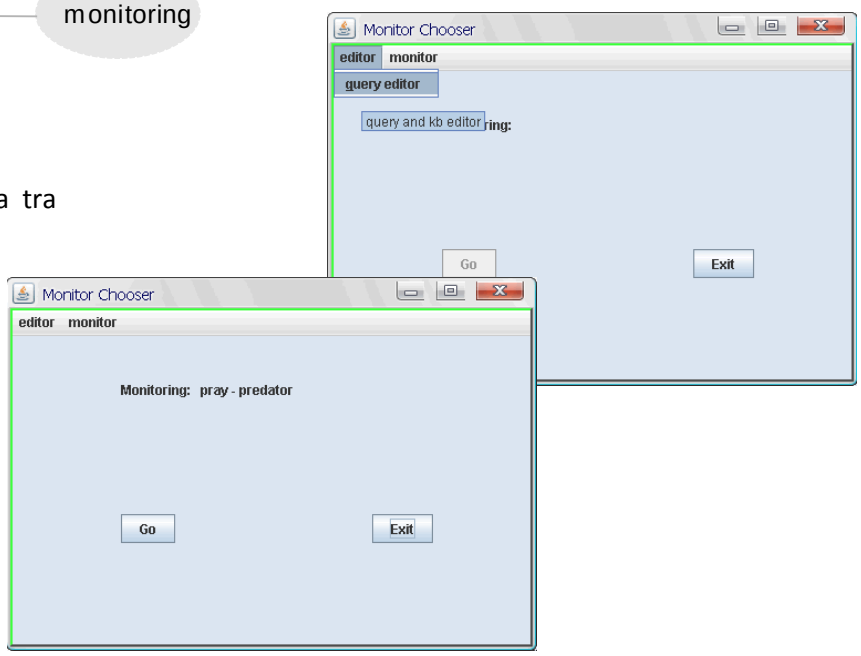

Figura 5-2 le interfacce grafiche del modulo di presentazione

alcune finestre di dialogo atte alla definizione e creazione delle componenti della *Knowledge Base*: un RoleDialog, un ConceptDialog, un QuantifierDialog ed infine un QueryDialog (Figura 5-4). L'utente dell'Editor esegue le seguenti azioni: inizialmente crea una nuova KB, vuota; in seguito alla creazione della KB divengono accessibili le finestre di dialogo per la creazione delle componenti del KB: creazione di ruoli, creazione di concetti, creazione di quantificatori. Combinando queste componenti, gli è poi possibile creare una nuova query tramite la QueryDialog (Figura 5-5).

È presente un input verifier che impedisce la creazione di una query non consentita dalle regole

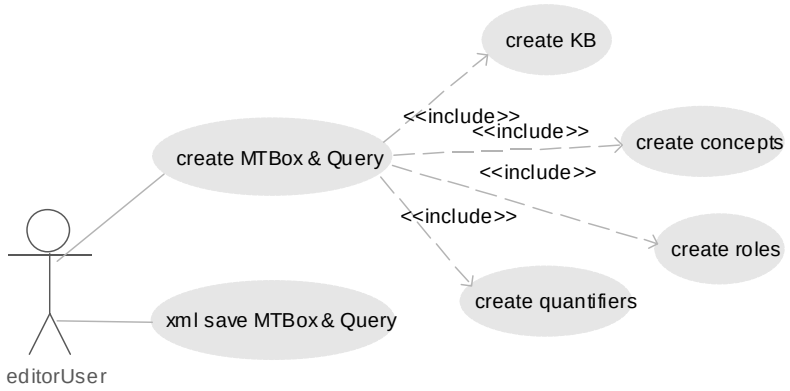

Figura 5-3 casi d'uso dell'editor: creare KB e query

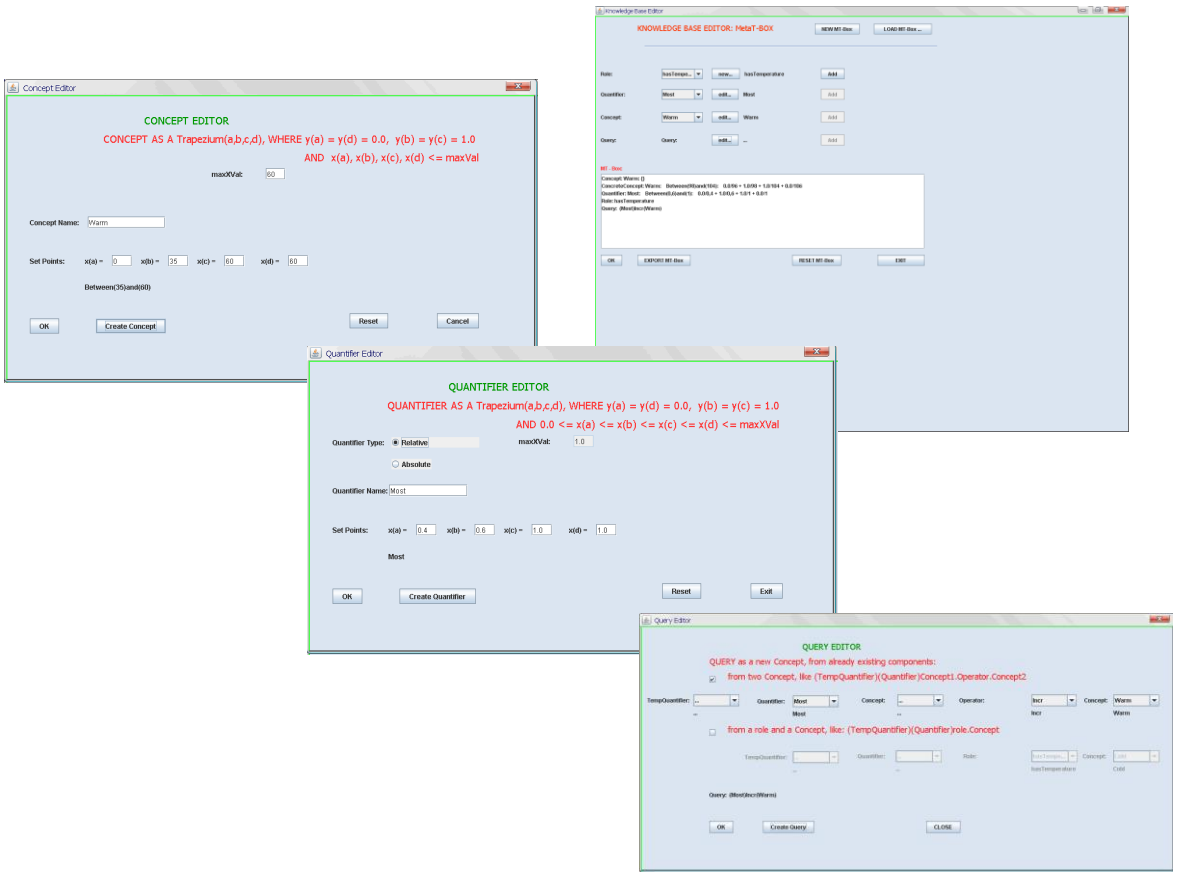

Figura 5-4 le interfacce grafiche dell'editor

sintattiche del linguaggio. Le query che si possono definire sono di due tipi: $(Q_T)(Q)R.C$ oppure $(Q_T)(Q)A(\mathrm{op})B$; il primo tipo corrisponde alla classe `RoleQuery` del package `reasoner`, come abbiamo visto: uno dei due quantificatori può anche essere nullo, mentre il ruolo R può essere nullo solo con Q nullo e Q_T non-nullo. Il secondo tipo corrisponde alla classe `OpQuery`: op corrisponde ad uno degli operatori binari (e allora A deve essere non-nulla) oppure ad uno degli operatori unari (e allora A deve essere nulla); Q è nullo per op = ⊓, ⊔; per op $= \mathcal{U}, \mathcal{S}$ o op del tipo

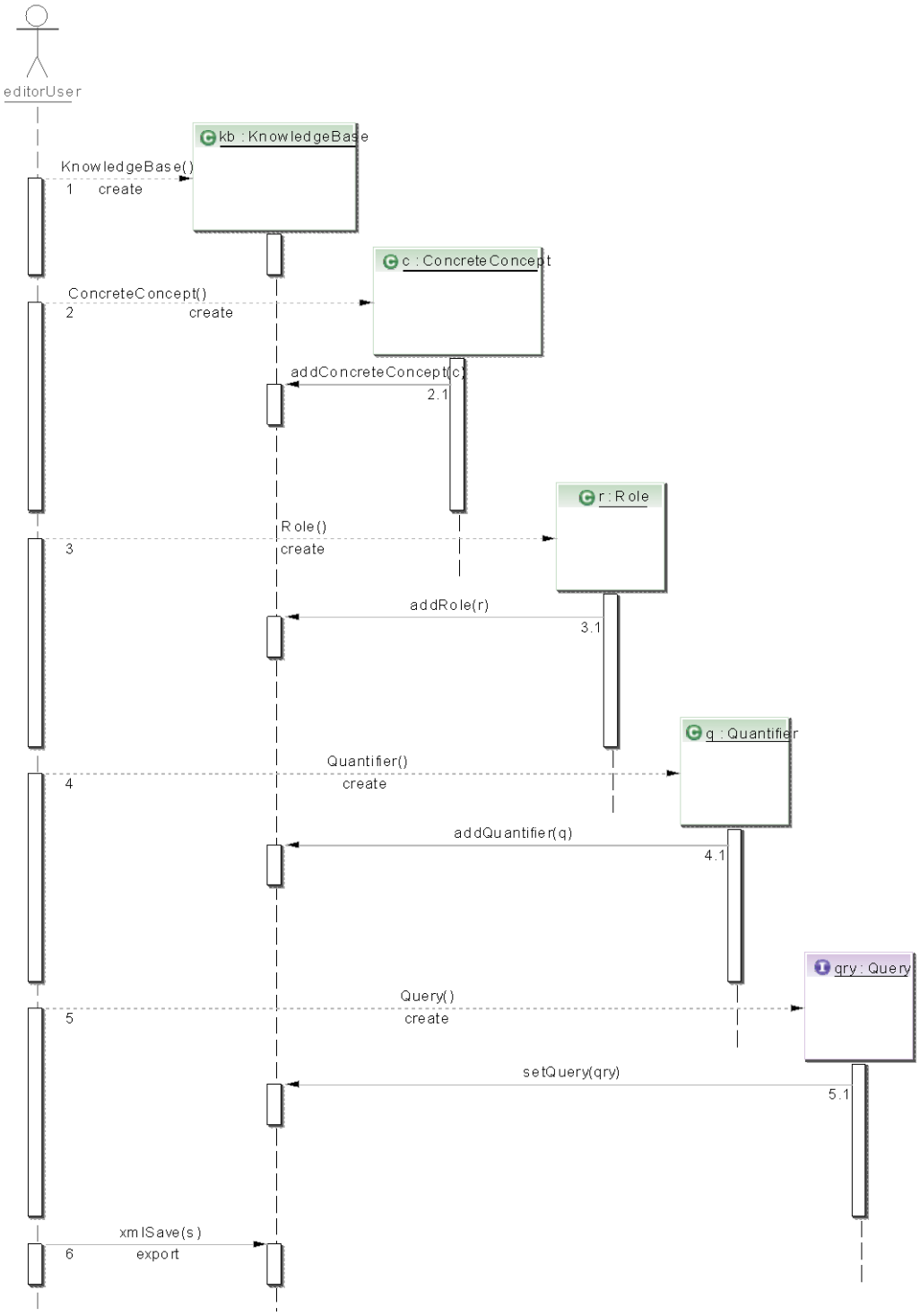

Figura 5-5 diagramma delle sequenze per l'Editor

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

operatore d'andamento, l'espressione $QA\mathrm{op}B$ è da intendersi una forma contratta di una delle due espressioni $(Q)ULine \sqsubseteq A\mathrm{op}B$ o $(Q)R.A\mathrm{op}B$, a seconda del contesto.

Una volta creata la query, all'utente dell'Editor è concesso esportare in formato xml l'*MTBox* creato. La Figura 5-5 costituisce un diagramma delle sequenze per l'Editor. L'implementazione dell'Editor segue il pattern MVC, come schematizzato dalla Figura 5-7. Il Model è realizzato dal package `reasoner` che ho descritto sopra. Ogni azione dell'Editor richiama un metodo corrispondente in `reasoner`. Ad esempio, per l'esportazione, è richiamato il metodo `xmlSave()`, che abbiamo visto. Il file esportato viene salvato in un repository (una cartella) di file xml, da dove potrà essere importato da parte delle applicazioni che vogliono utilizzarlo (ad esempio dalle applicazioni di monitoring che descriveremo).

5.3 Monitoring

Nel caso che l'utente abbia scelto di effettuare un monitoraggio, gli viene offerta l'opportunità di monitorare un sistema in tempo reale – nello specifico, un sistema di preda - predatori – oppure un sistema da database relazionale – nello specifico, le temperature di alcune località oppure la pressione arteriosa di alcune persone (Figura 5-6). Tutti e tre i monitor richiedono all'utente la scelta ed il caricamento della KB e della query da una libreria di query in formato xml.

5.4 Monitoring in tempo reale

La caratteristica di entrambi i sistemi di monitoring è che essi importano l'*MTBox* in formato xml da un repository di file xml (locale o remoto), laddove è stata esportata dall'editor.

Caratteristica saliente del sistema di monitoring in tempo reale (*RTMonitor*) è che esso inoltre importa periodicamente e ripetutamente i dati sullo stato corrente da un altro file xml, contenuto nello stesso repository xml od altrove, ed esportato, altrettanto periodicamente, dall'emulatore del sistema in tempo reale. I dati sullo stato corrente andranno ad aggiornare l'*ABox* del *Knowledge Base*. In seguito ad ogni aggiornamento, si esegue una nuova valutazione della Query contenuta nell'*MTBox*.

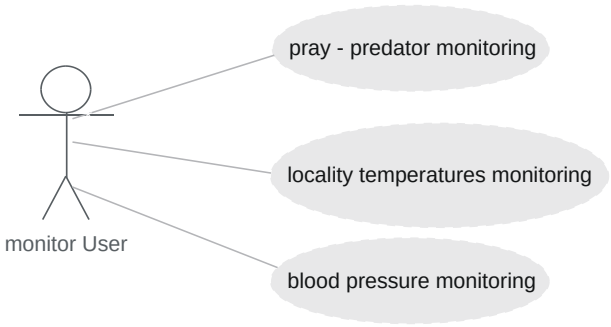

Figura 5-6 diagramma dei casi d'uso: scelta del monitor

Le operazioni periodiche di esportazione da parte dell'emulatore e importazione da parte del monitor sono del tutto trasparenti all'utente, il quale vede unicamente che, nella finestra grafica del monitor, i dati sullo stato corrente e i risultati della valutazione sono continuamente aggiornati.

5.4.1 Sistema prede – predatori

Come introduzione teorica, riprendiamo l'esempio 3 degli agenti automi alla ricerca di un obiettivo ed estendiamolo come un sistema di prede e predatori.

In prima approssimazione, supponiamo la posizione delle prede sufficientemente costante da poterle considerare *Static*. In seconda approssimazione, supponiamo che sia determinata e invariabile l'assegnazione di ogni predatore alla sua preda. La prima approssimazione sarà in seguito sciolta.

Le prede possono rappresentare obiettivi di diversa natura e dimensione, che per essere raggiunti necessitano di automi non umani, per motivi ambientali, di sicurezza o di dimensione. Si può

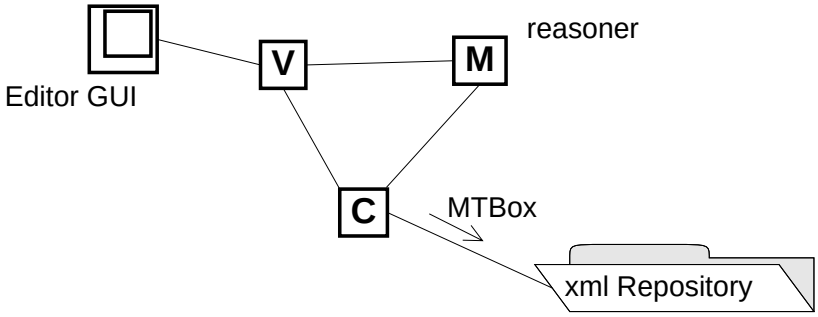

Figura 5-7 il pattern MVC per l'Editor

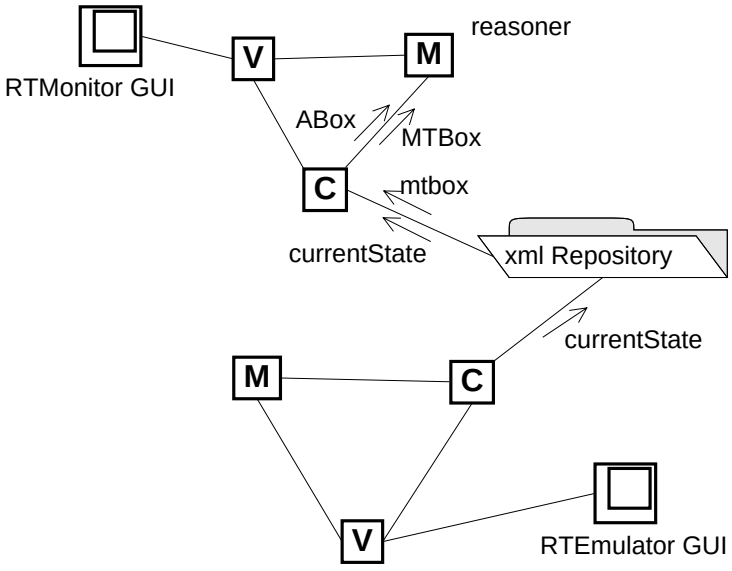

Figura 5-8 il pattern MVC per l'RTMonitor

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

pensare ad una chiazza di petrolio da dissolvere, ad una profondità sottomarina o una cavità vulcanica da esplorare, alle mine da eliminare, fino alle metastasi di un tumore nell'organismo umano da aggredire.

I predatori sono agenti automi il cui scopo è raggiungere il bersaglio e compiere in seguito qualche altra azione. L'azione presumibilmente sarà da compiere solo dopo che una determinata parte dei predatori abbiano raggiunto la preda.

La *Knowledge Base* contiene dunque, come abbiamo già visto, i seguenti concetti e ruoli (Figura 5-9):

$KB \supseteq \{Predator \sqsubseteq Dynamic\ ;\ distFromTarget : Predator \sqcap Event \rightarrow Distance\ ;\ Distance \sqsubseteq PositiveReal\ ;\ CloseDistance = Distance \sqcap LessThan(\sim 50)\ ;\ LessThan(\sim 50) = Trap\ (0, 0, 50, 60, 0, 500);\ Sometime \equiv Some_T\ ;\ Some \equiv \neg None = \neg Trap\ (0, 0, 0, 0)\}$. Ed, inoltre:

$position : Event \sqcup Static \rightarrow Position = Real \times Real$

$distance : Position \times Position \rightarrow PositiveReal$ (funzione algebrica)

$distance^{\mathcal{I}}(x, y) = |position(x) - position(y)|$ (norma)

$distFromTarget(x) = distance\ (x, hasTarget\ (x))$

$isTargetFor : Target \rightarrow Predator$

$hasTarget : Predator \rightarrow Target$

$hasTarget = isTargetFor^{-}$

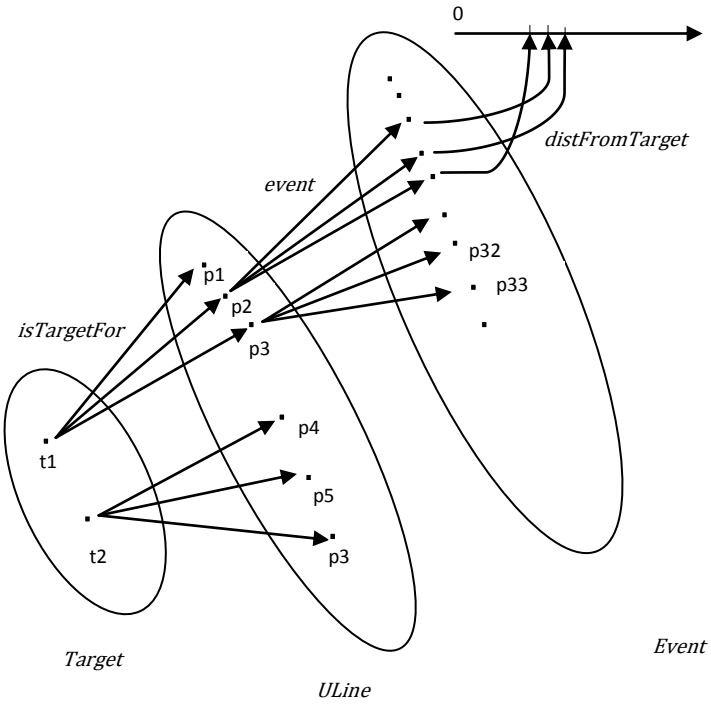

Figura 5-9 rappresentazione logica del sistema prede - predatori

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

ClosePredator $= \exists$ *distFromTarget.CloseDistance* è il concetto insieme dei predatori evento aventi distanza dal target 'vicina'.

SometimeClosePredator $=$ *ULine* $\sqcap$ *(Sometime)ClosePredator* è il concetto insieme dei predatori (linee universo) che sono stati almeno una volta a distanza dal target 'vicina'.

ApproachingPredator $= \nearrow$ *ClosePredator* è il concetto insieme dei predatori (linee universo) che si sono mediamente avvicinati. Da notare che una valutazione di questo tipo consente ai predatori di allontanarsi occasionalmente e seguire un percorso di avvicinamento non in linea retta, come può essere necessario in molti contesti.

ConstantClosePredator $=$ *SometimeClosePredator* $\sqcap \rightleftharpoons$ *ClosePredator* è il concetto insieme dei predatori in media costantemente vicini.

UsuallyClosePredator $=$ *(Usually)ClosePredator* $=$ *(Most*$_T$*) distFromTarget. CloseDistance* è il concetto insieme dei predatori solitamente vicini al target.

UsuallyInDangerTarget $=$ *(Most)isTargetFor.UsuallyClosePredator* è il concetto insieme degli obbiettivi la cui maggior parte dei predatori è solitamente vicina.

ConstInDangerTarget $=$ *(Most)isTargetFor.ConstantClosePredator* è il concetto insieme degli obbiettivi la cui maggior parte dei predatori è in media costantemente vicina.

Notiamo la differenza concettuale tra *UsuallyClosePredator* e *ConstantClosePredator*: il primo si ottiene quantificando sugli eventi a distanza vicina ed è quindi il risultato della valutazione congiunta del grado di appartenenza al concetto *ClosePredator* e della quantificazione *Most*. Il secondo non contiene quantificazione, ma è il risultato della sussunzione tra la presente appartenenza a *ClosePredator* e la futura o la passata.

Nella pratica, un *Predator* può essere *UsuallyClosePredator* con grado 1, se nella maggior parte dei suoi eventi è *ClosePredator*. Se nella maggior parte degli eventi si trova vicino all'obbiettivo, ma negli altri (ad esempio gli ultimi) si allontana sensibilmente il suo valore è 1. In tal caso, però, esso non può essere *ConstantClosePredator,* poiché tale concetto tiene conto delle deviazioni dalla costanza.

Che le prede siano *Static* è solo una prima approssimazione, per semplificare l'esempio. Nulla impedisce di considerarle *Dynamic* e allora il ruolo *distFromTarget* restituirà la distanza tra la posizione dell'evento predatore e la posizione del corrispondente evento preda, ad esso contemporaneo.

Considerando anche le prede *Dynamic*, chiamamole ora *Pray*, non più *Target* (Figura 5-10),

$Pray \sqsubseteq Dynamic$

$isPrayFor : Pray \rightarrow Predator$

$hasPray : Predator \rightarrow Pray$

$hasPray = isPrayFor^{-}$

foreach x: $Predator \sqcap Event$, $hasPray(x) = event \cdot hasPray \cdot uline(x) \sqcap concurrent(x)$

foreach x: $Pray \sqcap Event$, $isPrayFor(x) = event \cdot isPrayFor \cdot uline(x) \sqcap concurrent(x)$

$distFromPray : Predator \sqcap Event \rightarrow PositiveReal$

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

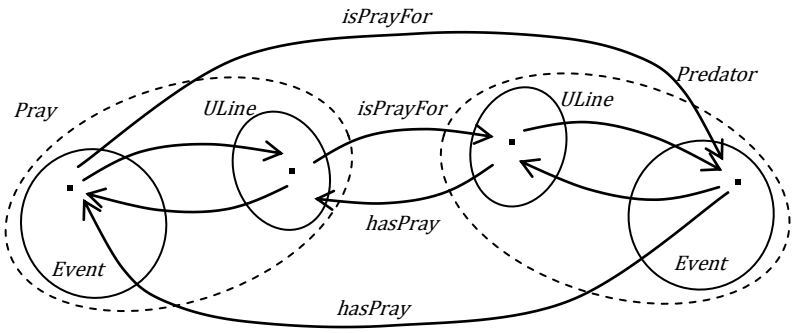

Figura 5-10 rappresentazione logica del caso in cui anche *Pray* è *Dynamic*

$position : Event \sqcup Static \rightarrow Position = Real \times Real$

$distance : Position \times Position \rightarrow PositiveReal$

$distance^{\mathcal{I}}(x, y) = |position(x) - position(y)|$

$distFromPray^{\mathcal{I}}(x) = distance^{\mathcal{I}}(x, (hasPray(x))$

$ClosePredator = \exists distFromPray.CloseDistance$

$ConstantClosePredator = (Sometime)ClosePredator \sqcap \rightleftharpoons ClosePredator$

$UsuallyInDangerPray = (Most)isPrayFor.(Usually)ClosePredator$

$ConstInDangerPray = (Most)isPrayFor.ConstantClosePredator$

Se, comunque, il nostro interesse è limitato alla distanza relativa tra preda e predatori, ci si può sempre ridurre ad un sistema in cui le prede siano individui *Static* e la posizione di ogni *Predator* sia misurata in un sistema di riferimento solidale con la preda corrispondente.

Il primo tipo di monitoraggio implementato si distingue dagli altri due. In esso infatti si valuta in tempo reale la query proposta, su un sistema che è attualmente in continua evoluzione. L'utente, scegliendo di monitorare un sistema preda – predatori, mette in esecuzione, prima di ogni altra cosa, un emulatore di tale sistema (Figura 5-11).

L'attività dell'emulatore è rappresentata dall'activity diagram di Figura 5-12. L'emulatore esegue un ciclo composto dalle seguenti fasi: rappresentazione grafica dello stato corrente, esportazione dello stato corrente in formato xml, aggiornamento dello stato. Nel caso di un comando di *scramble*, esegue uno scombussolamento random delle posizioni degli agenti dello stato corrente.

Lo stato corrente è costituito da un certo numero di agenti, prede e predatori e dalle loro posizioni rispetto ad un sistema di riferimento assoluto. L'aggiornamento dello stato corrente è implementato da un semplice algoritmo che mediamente avvicina i predatori alle corrispondenti prede, con oscillazioni variabili casualmente (in Figura 5-13 l'emulatore in esecuzione con un'unica preda e tre predatori). L'azione di 'scramble' allontana i predatori, casualmente, alle estremità del campo visibile. Le prede non sono immobili, ma anch'esse hanno un lento moto traslatorio da un lato all'opposto del campo visibile.

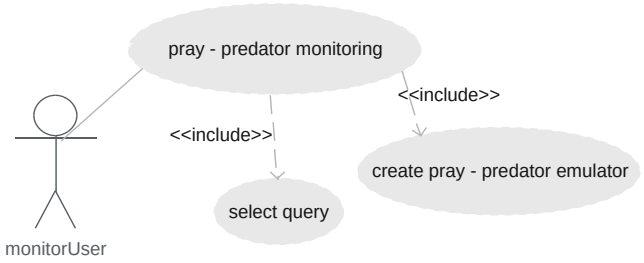

Figura 5-11 diagramma dei casi d'uso per il monitor pray - predator

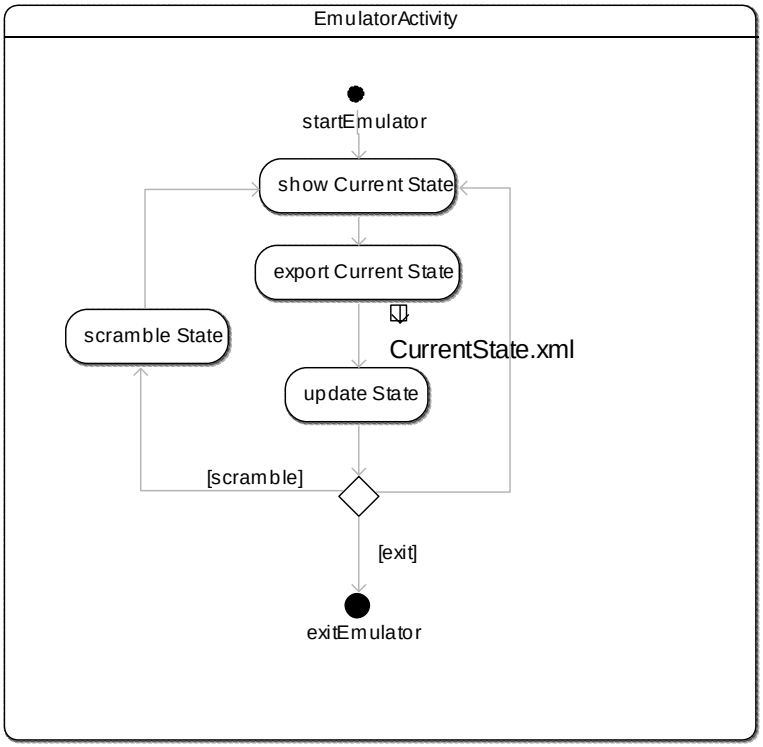

Figura 5-12 diagramma delle attività per l'emulatore del sistema pray-predator

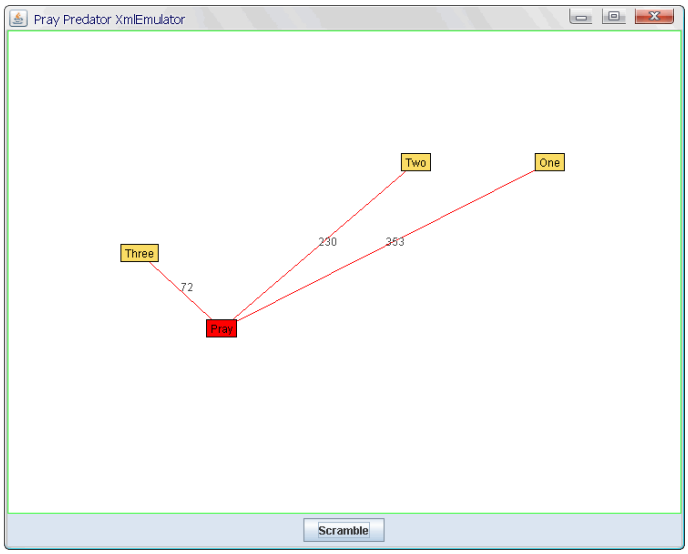

Figura 5-13 screenshot dell'emulatore

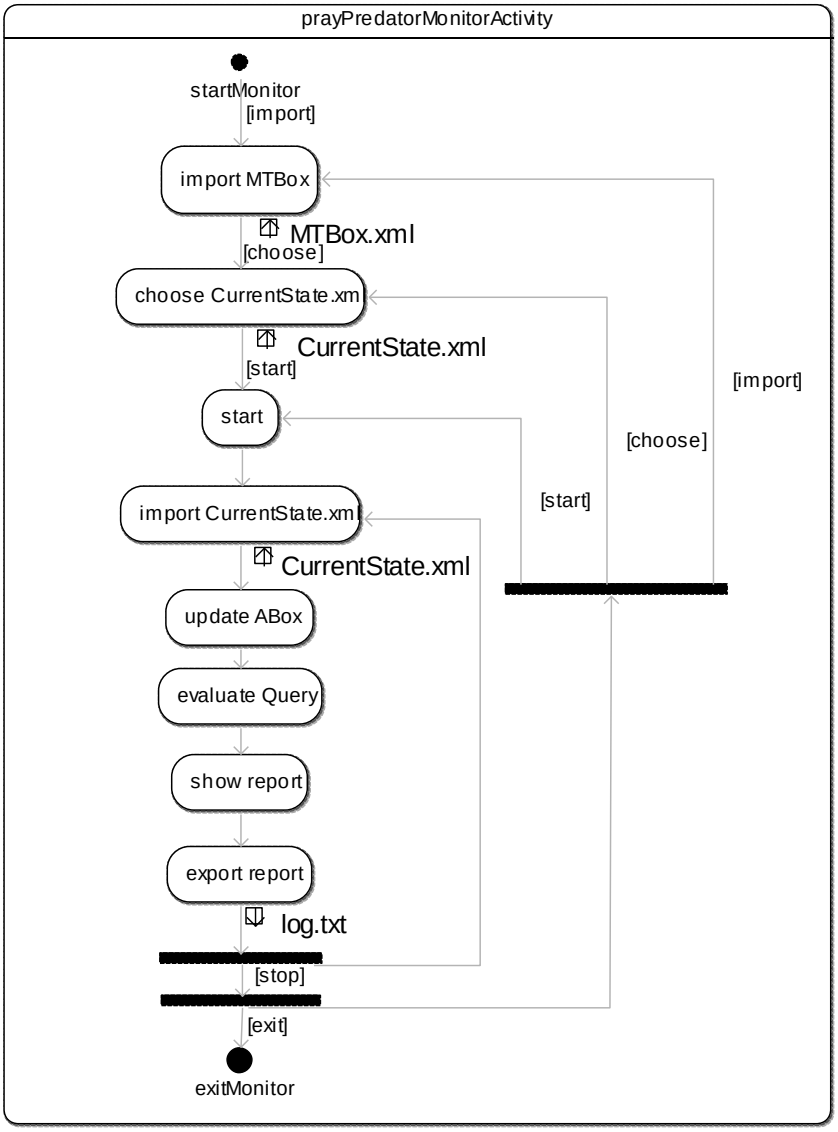

Figura 5-14 diagramma delle attività per il monitor in tempo reale

---

Lo stato corrente è esportato in un formato xml che si accorda con lo schema currentState.xsd, presente in Appendice, tra gli allegati. In esso si è definito un tipo complesso `CurrentStateType` avente due attributi, `name` di tipo `string` e `time` di tipo `dateTime`, che distinguono univocamente ogni istanza, e contenente una sequenza di elementi di tipo `NodeType`. Il tipo `NodeType` è caratterizzato da tre attributi: `name` di tipo `string`, `coord_x` e `coord_y` di tipo `double`. Tra gli allegati in Appendice si può vedere anche un esempio di esportazione nel file CurrentState.xml, in accordo con lo schema.

In corrispondenza ai due tipi `CurrentStateType` e `NodeType` sono state create le due classi java `CurrentStateType` e `NodeType`, ad esse legate tramite le librerie JAXB, con gli stessi meccanismi descritti a proposito dell'esportazione ed importazione dell'*MTBox*.

Il monitor importa periodicamente i dati sullo stato corrente dell'emulatore che è in esecuzione, con essi aggiorna l'*ABox* e sull'*ABox* aggiornato valuta nuovamente la query, in un ciclo che termina solo per decisione dell'utente. Laddove la query contenga operatori che ammettono

definizione semantica iterativa incrementale, il monitor esegue algoritmi di valutazione incrementali, che consentono maggior efficienza computazionale.

Un'activity diagram per il monitor in tempo reale è presente in Figura 5-14. In esso si vedono quali sono le principali attività svolte. In seguito alla decisione di 'import', viene importato il file, contenente l'*MTBox* e la query da valutare, e tradotto nelle classi del *KnowledgeBase* appartenenti al modulo `reasoner`. Successivamente viene scelto il file contenente lo stato corrente e, alla decisione di 'start', il sistema inizia il ciclo di monitoraggio, composto dai seguenti passi: importazione del file contenente lo stato corrente, traduzione dei dati in esso contenuti in *ABox*, valutazione della Query, esposizione di un report con i risultati della valutazione ed esportazione di tale report in un file di testo; infine, se non è stato richiesto uno 'stop', ritorno all'inizio del ciclo ed importazione del nuovo file di stato corrente.

Se invece è stato richiesto lo 'stop', a seconda della richiesta successiva, il sistema può reimportare un nuovo *MTBox* oppure scegliere un diverso file di stato corrente oppure ripartire con il monitoraggio oppure ancora può uscire e chiudere.

Gli agenti *Predator* sono *Dynamic*. Alla prima importazione del file CurrentState.xml, la business logic del monitor crea per ogni *Predator*, utilizzando il modulo `reasoner`, un'istanza della classe `ULineInd` avente il nome del *Predator* e un'istanza della classe `ULineEvent` ad esso corrispondente. Le prede invece sono considerate *Static*, in accordo con le considerazioni precedenti: ci basta considerare le distanze relative tra preda e predatori, quindi ogni preda è al centro del proprio sistema di riferimento.

Ad ogni importazione, compresa la prima, il monitor crea, per ogni agente *Predator*, una nuova istanza della classe `EventInd`; gli assegna il nome del *Predator* , il time dello stato corrente e la posizione dell'agente nello stato corrente e la inserisce nella corrispondente classe `ULineEvent`. Inoltre, valuta l'appartenenza dell'`EventInd` appena creato ai ruoli e concetti presenti nel *TBox* e li aggiorna con esso.

Alla fine dell'aggiornamento dell'*ABox*, il monitor chiede al modulo `reasoner` la valutazione della query; ne riceve i risultati e li mostra all'utente come testo formattato in una text area dell'interfaccia grafica. Gli stessi risultati sono poi esportati accodandoli in un file di log, da cui l' utente può visionare l'evoluzione del sistema. Uno screenshot del monitor in tempo reale è visibile in Figura 5-15.

5.5 Monitoring da DBMS relazionale

Il monitoring da DBMS relazionale importa i dati da monitorare da un DBMS relazionale e li traduce, utilizzando il modulo `reasoner`, in *ABox*.

Come nel caso del monitoring in tempo reale, i dati per la creazione dell'*MTBox* sono importati da un repository di file xml. In Figura 5-16 uno schema di come è realizzato il pattern MVC nel caso del monitor da DB relazionale.

Il modulo Controller è un'implementazione del pattern DAO (Data Access Object), ovvero è un oggetto che si occupa della connessione con il database e dell'esecuzione delle query sql su di esso.

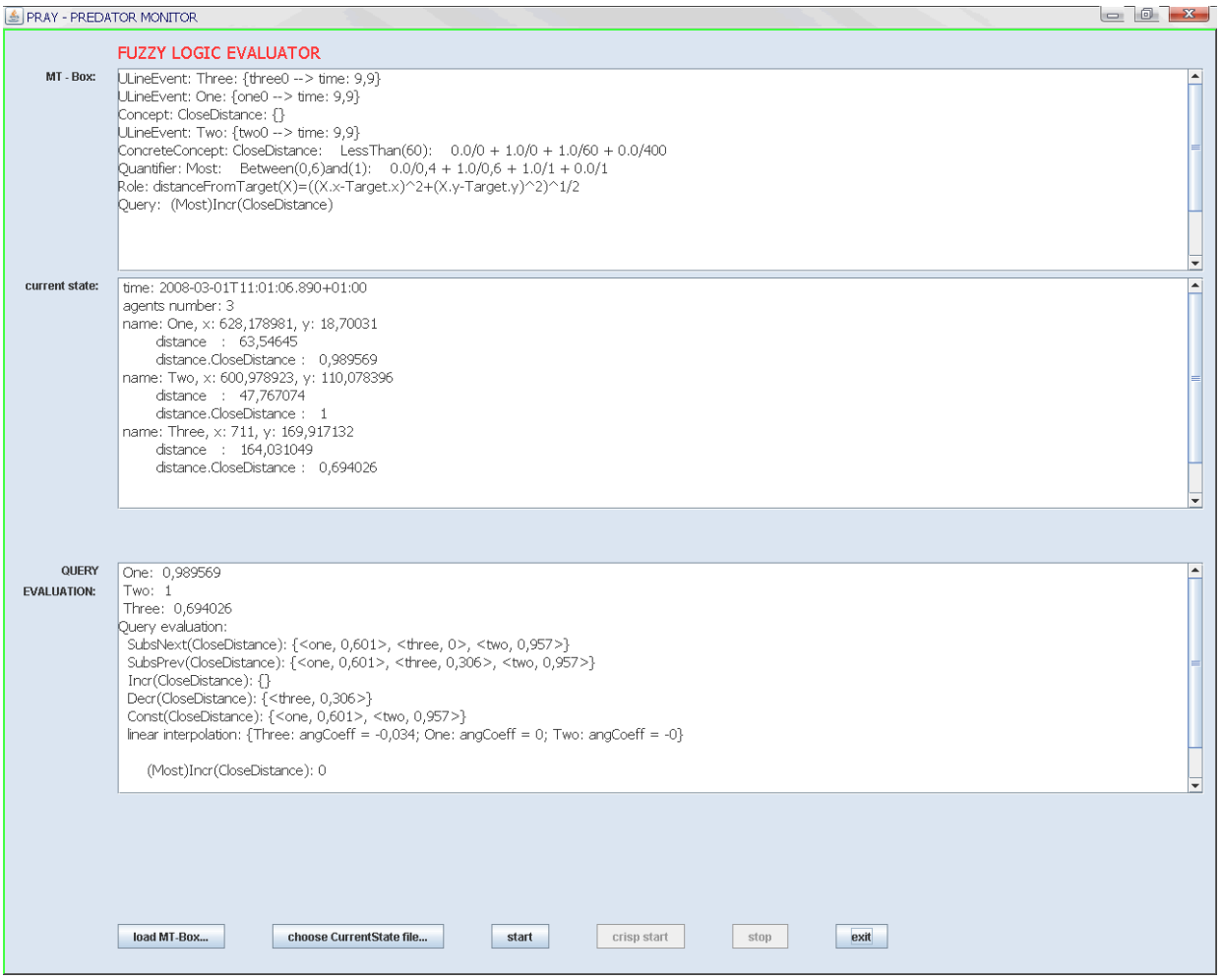

Figura 5-15 screenshot del monitor in tempo reale

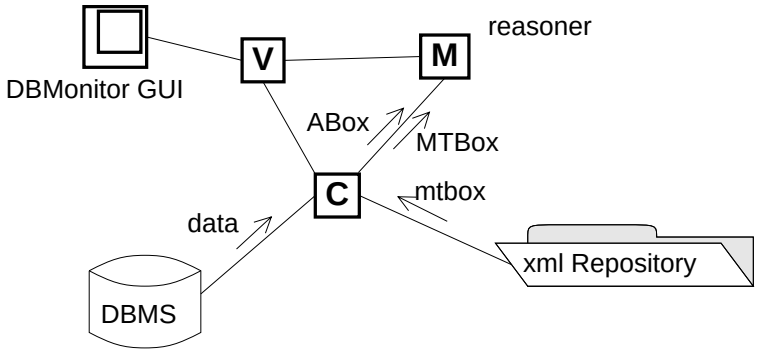

Figura 5-16 il pattern MVC per il monitor da DBMS

Esso è costituito da due classi (Figura 5-17). Una superclasse, denominata `Database`, gestisce un DBMS relazionale generico, stabilendo la connessione ed eseguendo query sql di select, update, insert e delete e restituisce le tabelle come `Vector` di `String`.

La sottoclasse `Db2Kb` è specializzata nella conversione dei dati ricevuti da una tabella di database in *ABox* per il `KnowledgeBase` di cui possiede il riferimento. Più in dettaglio, supponiamo che le tabelle abbiano uno schema per cui la prima colonna contenga il nome della tupla, di tipo `String`, e le successive colonne contengano valori numerici. Nella Tabella 5-1 è visibile un

Tabella 5-1 esempio di schema di tabella per il MonitoringDB

City	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982

frammento di uno schema di tabella, dove il primo campo, di nome 'City', contiene i nomi di città e gli altri campi contengono i valori medi della temperatura in quella città, nell'anno dato dal nome del campo.

Allora, si può interpretare ogni tupla come rappresentante un individuo City, inteso come `ULineInd`. Ogni singolo elemento della tupla rappresenta un individuo evento `EventInd`, nell'anno dato dal nome della colonna cui appartiene l'elemento e con la temperatura media misurata in quell'anno.

Per ogni tupla della tabella, il metodo `setABox()` della classe `Db2Kb` crea un'istanza della classe `ULineEvent` e le assegna il nome della tupla. Inoltre, per ogni valore numerico, crea un individuo `EventInd` e gli assegna il time dato dal nome della colonna. Per ogni ruolo e concetto presente nel *TBox* del `KnowledgeBase`, si valuta il grado d'appartenenza dell'`EventInd`, attraverso le membership functions, e si inserisce questo `EventInd` nel ruolo o concetto, con il grado d'appartenenza ottenuto.

Il modulo di View è costituito da un'interfaccia grafica analoga a quella di monitoring in tempo reale. In questo caso, però, al suo centro trova posto una regione grafica atta alla rappresentazione per punti dei valori importati dal DB relazionale.

Questo tipo di monitoraggio ben si adatta alla valutazione di dati la cui frequenza di campionamento e d'aggiornamento è bassa rispetto alla velocità di computazione. Nel primo esempio che considereremo, i dati sono aggiornati una volta all'anno; nel secondo presumibilmente non più di una volta all'ora. Se la frequenza dovesse divenire comparabile alla velocità di computazione, sarebbe più conveniente usare un monitor in tempo reale, come

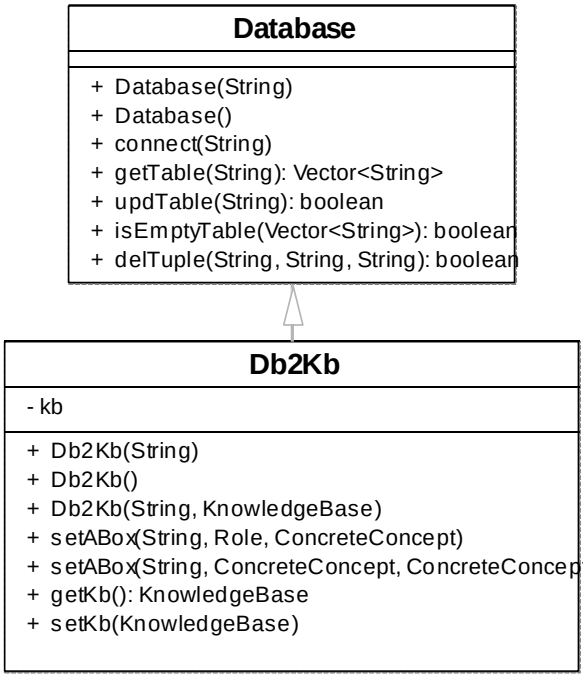

Figura 5-17 le classi del DAO

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

descritto nel paragrafo precedente.

5.5.1 Riscaldamento del pianeta?

Un tema di discussione assai acceso negli ultimi anni è se la temperatura globale del nostro pianeta sia in aumento oppure no. Considerato il nostro pianeta come un individuo *Static* che contiene le località di cui monitoriamo le temperature, località che invece sono da considerare individui *Dynamic*, il pianeta si riscalda se la maggior parte delle località ha temperatura media crescente nel tempo. Riprendo ed estendo i concetti esaminati nell'esempio 4, che ho propagato lungo le diverse fasi della discussione di questa tesi.

In questo caso, il valore della temperatura può essere qualunque (monitoriamo sia località calde che località fredde), ciò che importa è valutare se sia crescente.

È opportuno definire un concetto *Temp* linearmente crescente con la temperatura (Figura 5-18):

$$Temp = Trap\,(-40, 60, 60, 60, -40, 60)$$

$$Locality \sqsubseteq Dynamic$$

Per ogni località è definito il ruolo funzionale

$$hasTemperature : Event \sqcap Locality \rightarrow Temperature$$

$$fun\,(hasTemperature)$$

$$Temperature \sqsubseteq [-40.0, 60.0] \sqcap Real$$

Se definiamo

$$TempLocality = \exists hasTemperature.Temp$$

il concetto che associa ad ogni *Event* di *Locality* il grado di *Temp* della sua temperatura, allora le località a temperatura crescente, decrescente o costante apparterranno ai concetti, rispettivamente:

$$IncrTempLocality \equiv \nearrow TempLocality$$

$$DecrTempLocality \equiv \searrow TempLocality$$

$$ConstTempLocality \equiv \rightleftharpoons TempLocality.$$

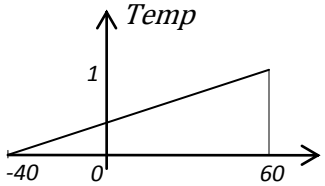

Figura 5-18 il concetto *Temp*: membership function

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Se ad ogni regione geografica è associato il ruolo di contenimento delle località:

$$hasLocality : Region \rightarrow Locality ,$$

allora le regioni in riscaldamento saranno descritte dal concetto

$$HeatingRegion = (Most)hasLocality.\nearrow TempLocality .$$

Se consideriamo il pianeta come un'unica grande regione, il riscaldamento globale può essere descritto col concetto:

$$GlobalHeating = (Most)hasLocality.\nearrow TempLocality .$$

In Figura 5-19 si può vedere uno screenshot dell'interfaccia grafica del monitor, dopo aver importato da un DB relazionale i dati sulle temperature medie di quattro località e dopo aver effettuato la valutazione di una Query. I dati utilizzati sono unicamente un esempio numerico di fantasia.

Nel caso particolare della figura, la Query valutata è

$$(Most)Incr(Temp) = (Most)hasLocality.\nearrow hasTemperature.Temp,$$ con

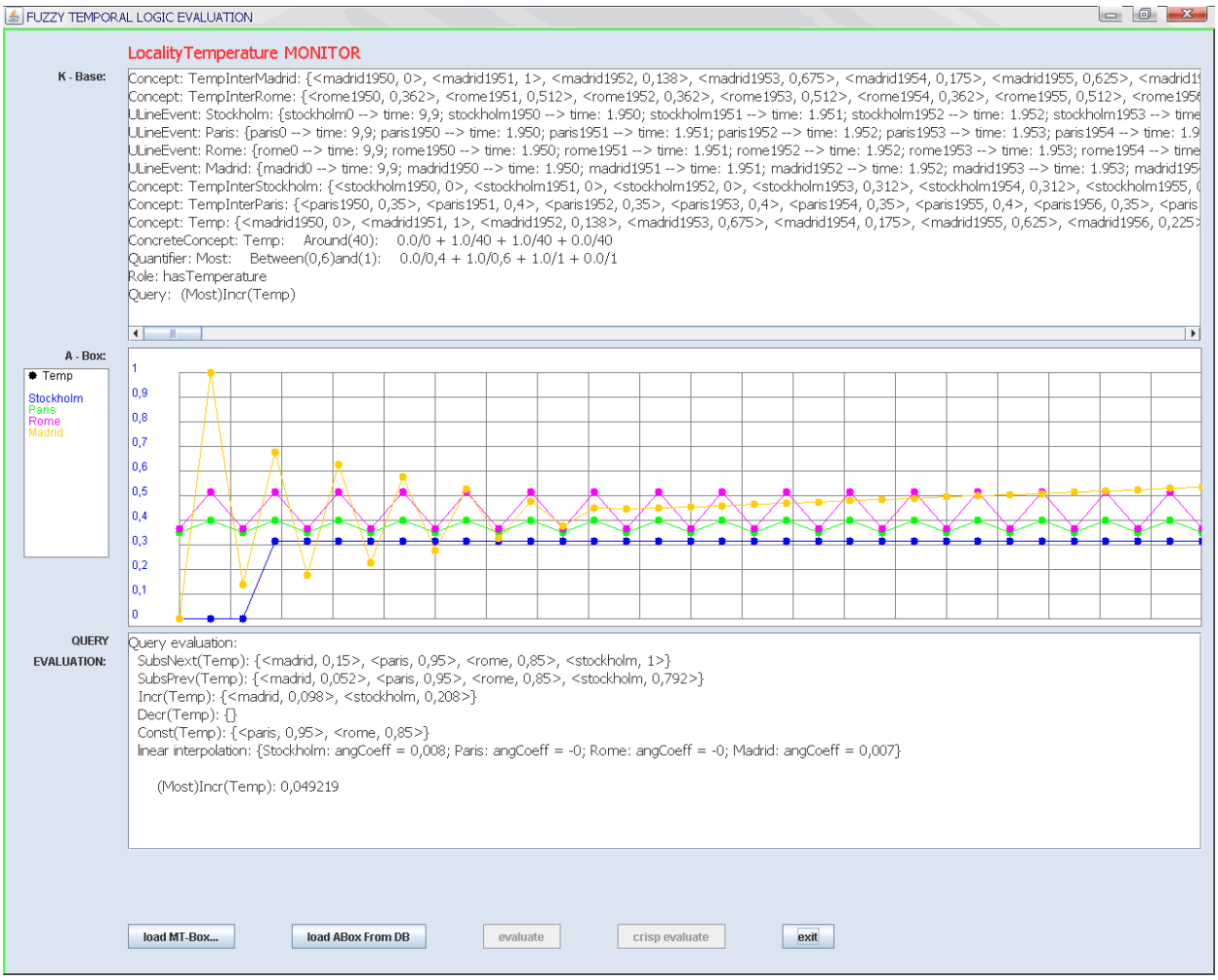

Figura 5-19 screenshot del monitorDb per le temperature di località

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Most = *Trap* (0.4, 0.6, 1.0, 1.0) e *Temp* = *Trap* (0, 40, 40, 40, 40).

Osserviamo che il monitor ci restituisce che i due individui di località stockholm e madrid appartengono al concetto *Incr(Temp)*, cioè i loro valori d'appartenenza a *Temp* sono stati valutati crescenti. Dal grafico vediamo infatti che i valori di stockholm (in blu) sono strettamente crescenti, mentre i valori di madrid sono inizialmente oscillanti, ma mediamente crescenti. Ne risulta un valore d'appartenenza a *Incr(Temp)* per stockholm maggiore di quello per madrid. Questi valori trovano conforto nei rispettivi coefficienti angolari calcolati per l'interpolazione lineare, positivi entrambi e uno maggiore dell'altro.

Le altre due località, paris e rome, sono state valutate appartenere al concetto *Const(Temp)*. Entrambe, infatti, hanno valori di *Temp* oscillanti, ma mediamente costanti (il loro coefficiente angolare per l'interpolazione lineare è nullo). Le oscillazioni sono più ampie per rome rispetto a paris e infatti il valore d'appartenenza di rome a *Const(Temp)* è inferiore di quello di paris.

Con questi valori dell'*ABox*, la valutazione della Query *(Most)Incr(Temp)* ha dato valore di verità = 0.049219.

Nella Tabella 5-2 sono riportati i risultati per alcune query significative, con lo stesso *ABox* e sempre intendendo, in questi casi, *(Most)C* = *(Most)hasLocality.C* .

Secondo questo campione di valori, dunque, si può affermare che, nonostante siano presenti località a temperatura media crescente, esse non sono la maggioranza; la maggior parte delle località ha invece temperatura media costante.

Si può notare altresì la differenza anche numerica tra i due concetti *Const(Temp)* e *(Usually)hasTemperature.Temp*. Il primo ammette come appartenenti quegli individui il cui valore nel concetto *Temp* è costante in media, con un valore d'appartenenza tanto maggiore quanto minori sono le oscillazioni dalla media. Il secondo contiene invece quegli individui che hanno solitamente (cioè per la maggior parte del tempo) valore non nullo in *Temp*, con un valore

Tabella 5-2 risultati per alcune query relative al monitoring di temperature di località

Query	risultati
(Most)Incr(Temp)	0.049219
(Most)Decr(Temp)	0.0
Const(Temp)	{<paris, 0,95>, <rome, 0,85>}
(Most)Const(Temp)	0.425
(Most)SubsNext(Temp)	0.9
(Most)SubsPrev(Temp)	0.779167
(Usually)hasTemperature.Temp	{<madrid : 0,479>, <paris : 0,371>, <rome : 0,426>, <stockholm : 0,312>}
ColdUntilWarm	{<madrid, 1>, <paris, 0,457>, <rome, 0,586>, <stockholm, 0,357>}

d'appartenenza dato da una media dei valori più frequenti.

5.5.2 Health care: pressione, glicemia, temperatura, …

E' possibile applicare i concetti e i ruoli introdotti anche in ambito sanitario, ove è importante valutare l'andamento nel tempo di determinati fattori fisiologici, quali la pressione arteriosa, la glicemia, il colesterolo, la temperatura ecc. E' cura del sanitario controllare che i valori di questi fattori si mantengano approssimativamente costanti e con piccole oscillazioni attorno ai valori considerati normali.

A titolo di esempio, possiamo limitarci alle valutazioni sulla pressione arteriosa. Inoltre, possiamo supporre che siano state somministrate diverse terapie sperimentali su un campione di pazienti e si voglia determinare quali di queste terapie siano efficaci, nel senso che riescano a mantenere i valori della pressione entro valori costantemente normali alla maggior parte dei pazienti che le hanno assunte.

Le terapie si descrivono come individui *Static*, mentre i pazienti si descrivono come *Dynamic* (Figura 5-20):

$$Therapy \sqsubseteq Static$$

$$Patient \sqsubseteq Dynamic$$

Le terapie vengono assunte dai pazienti; supponiamo che siano assegnate loro stabilmente, quindi come individui *ULine*. Ogni *Event* corrispondente ha una pressione da misurare, sia minima che massima:

$$takenBy : Therapy \rightarrow Patient \sqcap ULine$$

$$hasMinPressure : Patient \sqcap Event \rightarrow MinPressure$$

$$hasMaxPressure : Patient \sqcap Event \rightarrow MaxPressure$$

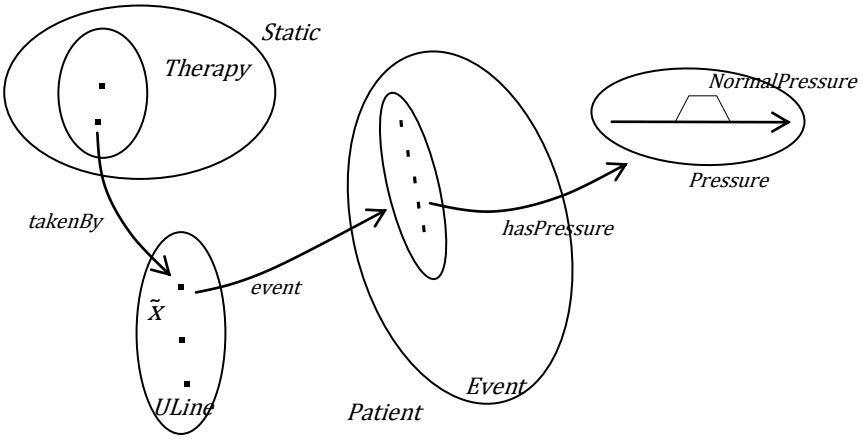

Figura 5-20 applicazione sanitaria: rappresentazione logica

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

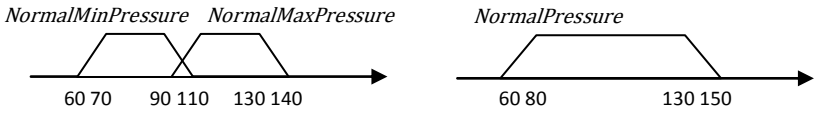

Figura 5-21 i concetti *NormalMinPressure, NormalMaxPressure, NormalPressure*

I concetti di pressione normale possono essere configurati con i due trapezi *NormalMinPressure* e *NormalMaxPressure* (Figura 5-21):

$$NormalMinPressure = Trap\,(60, 70, 90, 110, 0, 200)$$

$$NormalMaxPressure = Trap\,(90, 110, 140, 150, 0, 200).$$

Se vogliamo, possiamo anche semplificare e concentrare l'attenzione su un unico concetto

$NormalPressure = Trap\,(60, 80, 130, 150, 0, 200)$ e un unico ruolo

$hasPressure : Patient \sqcap Event \rightarrow Pressure.$

Allora, il concetto

$NormalPressurePatient = \exists hasPressure.NormalPressure$

rappresenta quei $Patient \sqcap Event$ aventi pressione normale, mentre il concetto

$ConstNormalPatient = (Some_T)\ NormalPressurePatient \sqcap \rightleftharpoons NormalPressurePatient$

rappresenta i pazienti con i valori di pressione mediamente costanti, entro la normalità.

Potremmo, dunque, descrivere efficaci quelle terapie per cui la maggior parte dei pazienti che le assumono ha una pressione che si mantiene costantemente nella norma:

$EffectiveTherapy = (Most)takenBy.ConstNormalPressurePatient.$

In Figura 5-22 è riportato uno screenshot dell'esecuzione di un monitor ad uso sanitario dopo la valutazione di una Query significativa, su valori di fantasia relativi a quattro pazienti.

Nel caso particolare della figura, la Query valutata è

$(Most)Const(NormalPressure)$, scrittura contratta equivalente a

$(Most)takenBy.\rightleftharpoons hasPressure.NormalPressure,$

avendo supposto di considerare un'unica terapia, con $NormalPressure$ definita come sopra e

$Most = Trap\,(0.4, 0.6, 1.0, 1.0).$

Nell'esempio valutato, l'individuo henry (in blu) ha un andamento in *NormalPressure* quasi sempre costante a 0.5, eccetto nei due ultimi valori che sono a 0.75. henry è stato valutato dunque appartenere al concetto $Incr(NormalPressure)$ con valore 0.35. Possiamo dire che "*la pressione arteriosa di* henry *ha avuto valori che si sono avvicinati alla norma*".

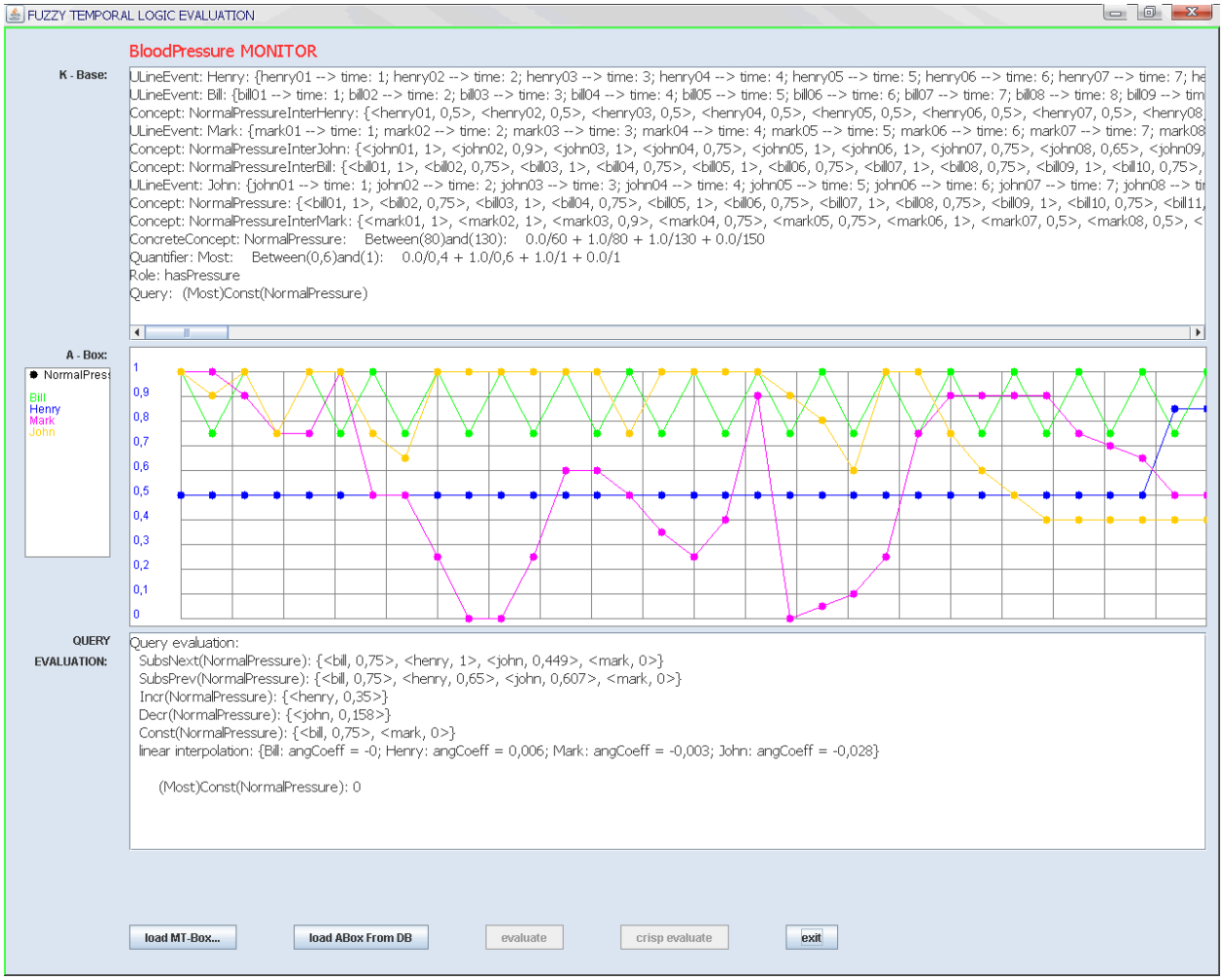

Figura 5-22 screenshot del monitorDb per health care

L'individuo bill (in verde) ha un andamento oscillatorio vicino a 1.0, con oscillazioni di ampiezza 0.25. bill è stato allora valutato appartenere al concetto *Const(NormalPressure)* con un valore 0.75. Possiamo dire che "*la pressione arteriosa di* bill *ha avuto valori che si sono mantenuti costantemente attorno alla norma*".

L'individuo john (in giallo) ha un andamento piuttosto irregolare nel concetto *NormalPressure*, con fasi di piena normalità (1.0) e cinque picchi di normalità lievemente diminuita. Nella fase finale, i valori scendono a 0.4 ed ivi rimangono. Questo andamento è stato valutato dal monitor appartenere al concetto *Decr(NormalPressure)*, con valore 0.157519. Possiamo dire che "*la pressione arteriosa di* john *ha avuto valori che sono leggermente diminuiti rispetto alla norma*".

Infine, i valori di pressione dell'individuo mark (in rosso) hanno un andamento molto irregolare, con forti oscillazioni dei corrispondenti valori nel concetto *NormalPressure*, tra 1.0 (norma) e 0.0 (fuori norma). Il monitor valuta questo andamento come appartenente al concetto *Const(NormalPressure)*, con valore 0.0, che si può tradurre nel linguaggio naturale come: "*la pressione arteriosa di* mark *non è né crescente né decrescente, dunque è mediamente costante; l'appartenenza al concetto che rappresenta la costanza media ridotta a zero* (*Const(NormalPressure)*(mark) = 0.0) *indica, però, che essa presenta oscillazioni d'ampiezza massima*".

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Concludendo, il monitor valuta che, con i dati dell'esempio analizzato, la query *(Most)Const(NormalPressure)* abbia valore di verità = 0.0, cioè che sia completamente falsa. Se questi fossero gli effetti della terapia assunta, si potrebbe desumere che la terapia sia completamente inefficace nel mantenere costante la pressione arteriosa.

Può interessare la valutazione di query differenti sugli stessi dati. Nella Tabella 5-3 sono riportati alcuni risultati.

Ricordo che il significato della query *(Usually)hasPressure.NormalPressure* = *Most event.hasPressure.NormalPressure* è il concetto contenente quegli individui *ULine* che "nella maggior parte del tempo" hanno pressione normale. Da ciò si può osservare che, effettivamente, l'individuo mark, con un andamento così irregolare da non appartenere né a *Incr* né a *Decr* né a *Const*, pur tuttavia ha grado d'appartenenza al concetto *(Usually)hasPressure. NormalPressure* maggiore dell'individuo henry, ovvero "solitamente ha la pressione normale" più che henry. Invece, come evidente, mark ha grado nullo per il concetto *(Always)hasPressure. NormalPressure* = *(All)event. hasPressure.NormalPressure*, cioè "è assolutamente falso che mark abbia sempre la pressione normale".

In conclusione, presento in Figura 5-23 un diagramma di deployment relativo all'intero progetto realizzato.

Tabella 5-3 risultati per alcune query relative al monitoring della pressione arteriosa

Query	risultati
(Most)Incr(NormalPressure)	0.0
(Most)Decr(NormalPressure)	0.0
(Most)Const(NormalPressure)	0.0
(Most)SubsNext(NormalPressure)	0.599501
(Most)SubsPrev(NormalPressure)	0.628261
(Usually)hasPressure.NormalPressure	{<bill, 0,894>, <henry, 0,5>, <john, 0,909>, <mark, 0,577>}
(Always)hasPressure.NormalPressure	{<bill, 0,75>, <henry, 0,5>, <john, 0,4>, <mark, 0>}

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

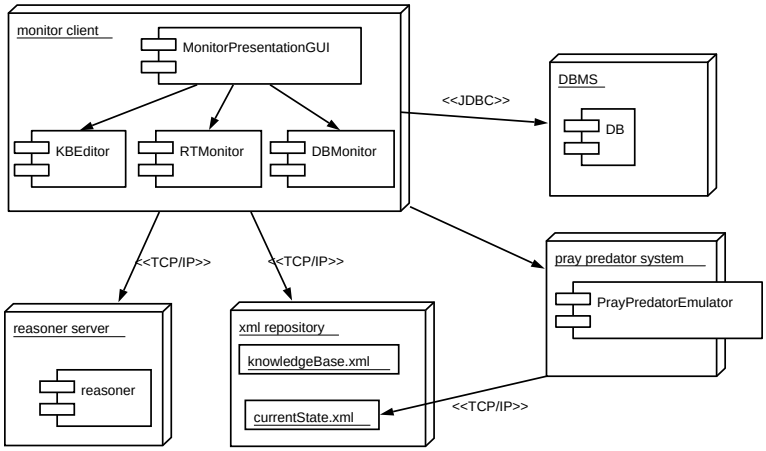

Figura 5-23 diagramma di deployment per l’intero progetto

6

Conclusioni e sviluppi futuri

In questa tesi ho voluto contribuire al progetto del *Semantic Web*.

Ho preso in considerazione i linguaggi logici DL fuzzy ed ho indagato la possibilità di estenderli in senso temporale secondo una prospettiva originale.

Adottando un punto di vista affine a quello della Fisica moderna, ho definito una *Upper Ontology* introducendo i concetti di *Linea Universo* ULine ed *Evento* Event. Un individuo, con un'esistenza la cui scansione nel tempo sia significativa, è rappresentato da una ULine. Una ULine contiene più Event. Ogni Event rappresenta l'individuo in un determinato istante temporale. Ho introdotto la dimensione temporale come un dominio concreto, sottoinsieme discreto di punti di $\mathbb{R}$. Un ruolo concreto, denominato *time*, associa ogni Event al proprio istante temporale.

Ho potuto conseguentemente definire una nuova semantica per gli operatori temporali tipici dei linguaggi temporali crisp – $\mathit{always}\ \Box$, $\mathit{sometime}\ \Diamond$, $\mathit{Until}\ \mathcal{U}$, $\mathit{Since}\ \mathcal{S}$, $\mathit{Next}\ \oplus$, $\mathit{Previous}\ \ominus$ – ed, in aggiunta, definire ex novo alcuni altri operatori temporali, che ho chiamato *operatori d'andamento* o *tendency operators*. Essi sono gli operatori unari $\mathit{Incr}\nearrow$, $\mathit{Decr}\searrow$, $\mathit{Const}\rightleftharpoons$, derivati a loro volta dagli altri due nuovi operatori unari $\mathit{SubsNext}$ e $\mathit{SubsPrev}$.

Applicando gli *operatori d'andamento* ad un concetto C, si ottengono i *Concetti d'Andamento* IncrC o $\nearrow C$, DecrC o $\searrow C$, ConstC o $\rightleftharpoons C$. Essi rappresentano quegli individui ULine, il cui andamento nel tempo dei valori d'appartenenza al concetto C sia, rispettivamente, crescente, decrescente o costante. Il valore d'appartenenza ai *Concetti d'Andamento* dipende sia dal valore dell'incremento o decremento che dalla monotonicità od oscillazione dell'andamento. Caratteristica che rende interessanti i *Concetti d'Andamento* è che sono ottenuti tramite le semplici operazioni logiche di sussunzione e intersezione.

Ho definito un'altra terna di concetti, i *Concetti d'Andamento Monotòno*, i quali definiscono l'andamento crescente, decrescente o costante, con valore dipendente solo dalla monotonicità. Essi si ottengono tramite gli *operatori d'andamento monotòno* $\mathit{MonIncr}$ e $\mathit{MonDecr}$.

Quando nel linguaggio logico si ammettono i quantificatori fuzzy, l'espressività si arricchisce notevolmente e ciò mi ha permesso di definire i nuovi *quantificatori temporali* fuzzy o *temporal*

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

quantifiers, quali *Usually*, *Often*, ecc., corrispondenti agli avverbi temporali dei linguaggi naturali. Anch'essi sono impiegati nella costruzione di nuovi concetti con semantica temporale.

Ho enunciato e dimostrato due teoremi a supporto della coerenza del sistema e un terzo teorema di utilità pratica.

In definitiva, la mia attenzione è stata rivolta alla traduzione in linguaggio logico delle espressioni del linguaggio naturale che contengono il divenire, espressioni che potrei classificare nei seguenti tre tipi.

1. Divenire come transizione da un concetto C_1 ad un altro concetto C_2.

   Per esse ho dato una nuova definizione semantica ai due operatori binari $Until\,\mathcal{U}$ e $Since\,\mathcal{S}$.

   Esempi di espressioni:

   $Arricchito = Povero\,\mathcal{U}\,Ricco$

   $DipendenteArricchito = Ricco\,\mathcal{S}\,Assunto$

2. Divenire come andamento (crescente, decrescente o costante) nell'appartenenza ad un concetto C.

   Per esse ho definito i nuovi operatori unari *SubsNext* e *SubsPrev* e gli *operatori d'andamento* $Incr\nearrow$, $Decr\searrow$, $Const\rightleftharpoons$.

   Esempio d'espressioni:

   $LocalitàSemprePiùCalda = \nearrow WarmLocality = \nearrow hasTemperature.Warm$

3. Quantificazione della frequenza o del numero di occorrenze nell'appartenenza ad un concetto C.

   Per esse ho definito i nuovi *quantificatori temporali* Q_T. I quantificatori temporali, in due casi particolari, equivalgono agli operatori unari *always* $\Box$ e *sometime* $\Diamond$, ai quali avevo dato definizione indipendente.

   Esempio d'espressioni:

   $LocalitàSpessoSoleggiata = (Often)SunnyLocality = (Often)hasWeather.Sunny$

   $BuonCliente = (Circa2voltesu3)\ \ acquista\ .\ ProdottiCostosi$

Ho mostrato come, in un sistema temporale così descritto, si possano definire anche altri tipi di query del tutto originali.

Ho implementato ed eseguito simulazioni numeriche e monitoraggi in tempo reale su sistemi in evoluzione temporale, aventi frequenza di campionamento e frequenza d'aggiornamento delle valutazioni paragonabile alla velocità del *reasoning*, ed altri monitoraggi, invece, su sistemi la cui frequenza di campionamento ed aggiornamento è relativamente bassa. In tutti i casi, le query eseguite danno risultati in buon accordo con le aspettative teoriche.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

In particolare, nei casi di query contenti *Concetti d'Andamento*, all'esecuzione della query si è affiancata l'esecuzione di un'interpolazione lineare degli stessi dati, al fine di offrire l'opportunità di un confronto qualitativo dei risultati.

Si constata che, effettivamente, individui appartenenti al concetto $ConstC$ hanno un insieme di valori in C, la cui interpolazione lineare restituisce un coefficiente angolare uguale o molto vicina allo zero; individui appartenenti al concetto $IncrC$ hanno valori in C la cui interpolazione lineare restituisce un coefficiente angolare positivo e, analogamente, individui appartenenti al concetto $DecrC$ hanno coefficiente angolare negativo. Inoltre, maggiore è l'ampiezza delle oscillazioni, cioè delle deviazioni dalla monotonicità, minore risulta il grado d'appartenenza ai rispettivi *Concetti d'Andamento*.

Tutto ciò può essere senza dubbio accademicamente interessante, ma può essere anche utile tecnologicamente. Nell'ambito del *Semantic Web*, numerose sono le situazioni in cui il poter valutare, tramite puro *reasoning* logico automatico, gli incrementi o i decrementi porterebbe un gran vantaggio. Ho presentato, nel corso della tesi, diversi esempi in campo commerciale, sanitario, ambientale e di controllo automatico.

Numerosi sono gli aspetti che sarebbe interessante investigare in ricerche future.

- Sarebbero da affrontare i problemi sulla complessità del calcolo per un linguaggio esteso in tal modo, in vista della progettazione ed implementazione di un reasoner più completo.
- Si potrebbe concretamente pensare all'estensione del linguaggio OWL, per comprendere nella *MetaBox* i nuovi operatori temporali e la definizione della funzione parametrica trapezoidale $Trap()$. Riflettere se esistono inconvenienti logici od implementativi nel consentire la definizione dei Quantificatori fuzzy nella *TBox*, assieme ai Concetti Concreti, o se piuttosto essi debbano restare nella loro sede naturale del *MetaBox*.
- Il linguaggio logico si arricchirebbe ulteriormente se valutazioni di complessità consentissero l'introduzione di operatori temporali dipendenti dalla lunghezza τ dell'intervallo temporale: $\mathcal{U}_\tau$, $\mathcal{S}_\tau$, $\Diamond_\tau$, $\Box_\tau$ $Q_{T\tau}$, ecc.

  Un esempio di definizione semantica potrebbe essere

$$(\Diamond^{-}{}_{\tau} C)^{\mathcal{I}}(\mathrm{x}) = \begin{cases} C^{\mathcal{I}}(\mathrm{x}); & \text{per } \mathrm{x} \in Static \\ \sup_{\substack{\mathrm{x}_1 \in event(\mathrm{x}) \wedge \\ 0 \le \_now\_-time(\mathrm{x}_1) \le \tau}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per } \mathrm{x} \in ULine \\ \sup_{\substack{\mathrm{x}_1 \in event(uline(\mathrm{x})) \wedge \\ 0 \le time(\mathrm{x}) - time(\mathrm{x}_1) \le \tau}} \{C^{\mathcal{I}}(\mathrm{x}_1)\}; & \text{per } \mathrm{x} \in Event \end{cases} .$$

  Si potrebbero avere concetti corrispondenti ad espressioni del tipo: "arricchito in tre anni", "temperatura cresciuta negli ultimi trent'anni" , "temperatura costante in un arco di duecento anni". Per altra via, si potrebbero ottenere le stesse opportunità consentendo al concetto concreto $Time$ di avere sottoconcetti.
- Sarebbe da affrontare la questione che, come visto, l'operazione *Next* $\oplus$ priva il concetto di un individuo, l'ultimo, e *Prev* $\ominus$ priva il concetto del primo individuo. Se eseguo in successione le due operazioni sul concetto C, $\oplus\ominus C$ oppure $\ominus\oplus C$, non ottengo C, come sarebbe legittimo aspettarsi, ma C privato del primo e dell'ultimo individuo. La composizione dei due operatori è diversa dall'identità, mentre sarebbe naturale che si potesse considerare l'uno l'inverso dell'altro. Per ovviare al problema, si potrebbe dotare il concetto di un campo aggiuntivo opzionale che contenga un riferimento all'individuo tagliato, oppure al concetto da cui esso stesso è derivato.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

- Assai interessante sarebbe anche l'introduzione di una o più dimensioni spaziali: essa si potrebbe affrontare con lo stesso approccio dell'estensione temporale, pur a costo di notevole aggravio sulla complessità. La sequenza degli *Event* corrispondenti ad una Linea Universo *ULine* non sarebbe più una retta (orientata), ma una linea vincolata ad un piano (n+1)-dimensionale, quando si inseriscano n dimensioni spaziali.

Riassumendo, i principali contributi originali in questa tesi possono elencarsi in: definizione di *quasi-sussunzione*; tutta la *Upper Ontology* temporale, con l'introduzione dei concetti di *Linea Universo ULine* ed *Evento Event* e concetti e ruoli connessi; introduzione del concetto *Time* come concetto concreto; nuova definizione fuzzy della semantica degli operatori *always* $\Box$, *sometime* $\Diamond$, *Until* $\mathcal{U}$, *Since* $\mathcal{S}$ (con definizioni weak, strong, inclusive ed esclusive), *Next* $\oplus$, *Previous* $\ominus$ (*Last*); i nuovi *operatori* e *Concetti d'Andamento SubsNext*, *SubsPrev*, *Incr* $\nearrow$, *Decr* $\searrow$ e *Const* $\rightleftharpoons$ e loro semantica; i nuovi *operatori* e *Concetti d'Andamento Monotòno MonIncr*, *MonDecr* e loro semantica; i nuovi *quantificatori temporali* fuzzy; tre teoremi con dimostrazione; analisi di numerosi tipi di nuove query ora possibili; progettazione ed implementazione di tutte le applicazioni dimostrative e documentazione.

7

Appendici

Questo capitolo contiene alcune appendici. In 7.1 enuncio e dimostro un teorema che non ho inserito nella parte teorica per non appesantirla, ma che consente maggior efficienza computazionale in implementazione. In 7.2 do un cenno brevissimo sul percorso di un reasoner nell'esecuzione di una query di sussunzione generale. In 7.3 si trova qualche brano di codice più significativo e documentazione.

7.1 Un terzo teorema

Teorema 3. Per ogni Concetto C e per ogni interpretazione $\mathcal{I}$, è vera la seguente uguaglianza:

$$(C \sqsubseteq \ominus C)^{\mathcal{I}} = (\oplus C \sqsubseteq C)^{\mathcal{I}} \qquad 7\text{-}1$$

Dimostrazione. (Valida anche per la '*quasi sussunzione*' $\tilde{\sqsubseteq}$).

Ricordiamo che con la sussunzione $A \sqsubseteq B$ si intende che

$\mathrm{supp}(A) = \mathrm{supp}(B)$ e

$A(\mathrm{x}) \leq B(\mathrm{x})$ per ogni $\mathrm{x} \in \mathrm{supp}(A) = \mathrm{supp}(B)$, 7-2

avendo indicato con $\mathrm{supp}()$ il supporto di un concetto.

Gli operatori $\oplus$ e $\ominus$, per definizione, equivalgono all'identità per individui *Static* o *ULine*:

$\oplus C^{\mathcal{I}}(\mathrm{x}) = \ominus C^{\mathcal{I}}(\mathrm{x}) = C^{\mathcal{I}}(\mathrm{x})$ per $\mathrm{x} : \mathit{Static} \sqcup \mathit{ULine}$,

dunque, per queste x si ha certamente $C^{\mathcal{I}}(\mathrm{x}) \leq \oplus C^{\mathcal{I}}(\mathrm{x})$ e $\oplus C^{\mathcal{I}}(\mathrm{x}) \leq C^{\mathcal{I}}(\mathrm{x})$.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Consideriamo allora le x: *Event*. Sappiamo esistere una partizione su *Event* , per cui ogni x:*Event* corrisponde ad una e una sola $\tilde{x}$: *ULine* tale che $\tilde{x} = uline(x)$. Possiamo, senza perdita di generalità, restringere l'attenzione sugli eventi x corrispondenti ad un'unica *ULine* $\tilde{x}$, sapendo che, se per essa la (7-1) sarà verificata, basterà ripetere il ragionamento per tutte le *ULine,* perché la (7-1) sia verificata nella generalità. Se $\tilde{x}$: *ULine*, $X \equiv event(\tilde{x})$ e $CX \equiv C \sqcap X$, la (7-1) diviene

$$(CX \sqsubseteq \ominus CX)^{\mathcal{I}} = (\oplus CX \sqsubseteq CX)^{\mathcal{I}} \qquad 7\text{-}3$$

Abbiamo già visto che $\oplus C^{\mathcal{I}}(x) = \bot$ per x:*lastEvent*($\tilde{x}$) e $\ominus C^{\mathcal{I}}(x) = \bot$ per x: *firstEvent*($\tilde{x}$), dove *lastEvent* e *firstEvent* sono i ruoli che associano la Linea Universo, rispettivamente, al suo ultimo e primo *Event*, in ordine temporale. Allora, se $supp(CX) = \{x_1, x_2, \ldots, x_n\}$, abbiamo $supp(\oplus CX) = \{x_1, x_2, \ldots, x_{n-1}\}$ e $supp(\ominus CX) = \{x_2, x_3, \ldots, x_n\}$. Le due sussunzioni della (7-1) hanno significato, dunque, trasformandole in

$$(CX \setminus firstX \sqsubseteq \ominus CX)^{\mathcal{I}} = (\oplus CX \sqsubseteq CX \setminus lastX)^{\mathcal{I}}, \qquad 7\text{-}4$$

dove, per comodità, ho definito $firstX \equiv firstEvent(\tilde{x})$, $lastX \equiv lastEvent(\tilde{x})$, $CX \setminus firstX \equiv C \sqcap (X \setminus firstX)$, $CX \setminus lastX \equiv C \sqcap (X \setminus lastX)$. E' dunque da dimostrare la (7-4).

Dal metodo *GD* (2-16) si ha:

$(CX \setminus firstX \sqsubseteq \ominus CX)^{\mathcal{I}} = GD(\ominus CX / CX \setminus firstX)$

$$= \sum_{\alpha_i \in \Lambda(\ominus CX / CX \setminus firstX)} (\alpha_i - \alpha_{i+1})\, All\left(\frac{\left|(CX \setminus firstX \sqcap \ominus CX)_{\alpha_i}\right|}{\left|(CX \setminus firstX)_{\alpha_i}\right|}\right);$$

$(\oplus CX \sqsubseteq CX \setminus lastX)^{\mathcal{I}} = GD(CX \setminus lastX / \oplus CX)$

$$= \sum_{\alpha_i \in \Lambda(CX \setminus lastX / \oplus CX)} (\alpha_i - \alpha_{i+1})\, All\left(\frac{\left|(CX \setminus lastX \sqcap \oplus CX)_{\alpha_i}\right|}{\left|\oplus CX_{\alpha_i}\right|}\right).$$

Dimostrerò dapprima che i due insiemi Λ sono uguali, poi che, per ogni $\alpha \in \Lambda$, i rapporti delle cardinalità sono uguali.

$$\Lambda(\ominus CX / CX \setminus firstX) = \Lambda(\ominus CX \sqcap CX \setminus firstX) \cup \Lambda(CX \setminus firstX); \qquad 7\text{-}5$$

$$\Lambda(CX \setminus lastX / \oplus CX) = \Lambda(CX \setminus lastX \sqcap \oplus CX) \cup \Lambda(\oplus CX). \qquad 7\text{-}6$$

Abbiamo:

$\Lambda(\oplus CX) = \{(\oplus CX)(x_i) \mid x_i \in supp(\oplus CX) = \{x_1, x_2, \ldots, x_{n-1}\}\} = \{(CX)(x_{i+1}) \mid x_i \in \{x_1, x_2, \ldots, x_{n-1}\}\} = \{CX(x_i) \mid x_i \in \{x_2, x_3, \ldots, x_n\}\} = \{CX(x_i) \mid x_i \in supp(CX \setminus firstX) = \{x_2, x_3, \ldots, x_n\}\} = \Lambda(CX \setminus firstX)$;

inoltre:

$\Lambda(\ominus CX \sqcap CX \setminus firstX) = \{\min[\ominus CX(x_i), CX \setminus firstX(x_i)] \mid x_i \in \{x_2, x_3, \ldots, x_n\}\} = \{\min[CX(x_{i-1}), CX(x_i)] \mid x_i \in \{x_2, x_3, \ldots, x_n\}\}$;

$\Lambda(CX \setminus lastX \sqcap \oplus CX) = \{\min[CX \setminus lastX(x_i), \oplus CX(x_i)] \mid x_i \in \{x_1, x_2, \ldots, x_{n-1}\}\} = \{\min[CX(x_i), CX(x_{i+1})] \mid x_i \in \{x_1, x_2, \ldots, x_{n-1}\}\} = \{\min[CX(x_{i-1}), CX(x_i)] \mid x_i \in \{x_2, x_3, \ldots, x_n\}\} = \Lambda(\ominus CX \sqcap CX \setminus firstX)$.

verso $f\_\mathcal{SHOIQ}^{+}\mathcal{T}$ (D)

Quindi, i due insiemi (7-5) e (7-6) sono uguali: li chiamerò Λ.

Ora mostrerò che, per ogni $\alpha \in \Lambda$, si ha che $\left(\frac{|(CX \setminus firstX \sqcap \ominus CX)_\alpha|}{|(CX \setminus firstX)_\alpha|}\right) = \left(\frac{|(CX \setminus lastX \sqcap \oplus CX)_\alpha|}{|\oplus CX_\alpha|}\right)$.

$$\frac{|(CX \setminus firstX \sqcap \ominus CX)_\alpha|}{|(CX \setminus firstX)_\alpha|} = \frac{|\{x_i \,|(CX \setminus firstX \sqcap \ominus CX)(x_i) \geq \alpha\}|}{|\{x_i \,|(CX \setminus firstX)(x_i) \geq \alpha\}|} = \frac{|\{x_i \,|min\,[(CX\,(x_i), CX(x_{i-1})] \geq \alpha, x_i \in \{x_2, \ldots, x_n\}\}|}{|\{x_i \,|CX(x_i) \geq \alpha, x_i \in \{x_2, \ldots, x_n\}\}|};$$

$$\frac{|(CX \setminus lastX \sqcap \oplus CX)_\alpha|}{|\oplus CX_\alpha|} = \frac{|\{x_i \,|(CX \setminus lastX \sqcap \oplus CX)(x_i) \geq \alpha\}|}{|\{x_i \,|\oplus CX\,(x_i) \geq \alpha\}|} = \frac{|\{x_i \,|min\,[(CX\,(x_i), CX(x_{i+1})] \geq \alpha, x_i \in \{x_1, \ldots, x_{n-1}\}\}|}{|\{x_i \,|CX(x_{i+1}) \geq \alpha, x_i \in \{x_1, \ldots, x_{n-1}\}\}|} =$$

$$\frac{|\{x_i \,|min\,[(CX\,(x_{i-1}), CX(x_i)] \geq \alpha, x_i \in \{x_2, \ldots, x_n\}\}|}{|\{x_i \,|CX(x_i) \geq \alpha, x_i \in \{x_2, \ldots, x_n\}\}|} = \frac{|(CX \setminus firstX \sqcap \ominus CX)_\alpha|}{|(CX \setminus firstX)_\alpha|}$$

q.e.d.

Il teorema si sarebbe potuto dimostrare anche con considerazioni dirette. Basta notare che 1) $C \setminus first$ contiene gli stessi valori di $\oplus C$, posticipati di un'unità temporale; 2) $\ominus C$ contiene gli stessi valori di $C \setminus last$, posticipati di un'unità temporale (Figura 7-1). Quindi, la relazione tra i valori di $C \setminus first \sqsubseteq \ominus C$ è identica alla relazione tra i valori di $\oplus C \sqsubseteq C \setminus last$.

Notiamo che non abbiamo fatto ipotesi o deduzioni sul quantificatore *All*. Se al suo posto vi fosse stato il quantificatore *AlmostAll*, la dimostrazione sarebbe stata ugualmente valida. Dunque, il teorema vale anche per la *quasi – sussunzione* $\tilde{\sqsubseteq}$.

7.2 Reasoning

Di fronte ad una Query di sussunzione del tipo $\mathcal{K} \models \langle A \sqsubseteq B, \mathrm{n} \rangle$? , il reasoner dovrebbe eseguire i seguenti passi.

1. n = 0;
2. Inferenza dagli assiomi di specializzazione del *TBox*.

Ricerca nel *TBox* $\mathcal{T}$: se dagli assiomi di specializzazione di $\mathcal{T}$ si può inferire

$\mathcal{T} \models \langle A \sqsubseteq B, \geq \mathrm{m} \rangle$ allora $\mathrm{n} = max(\mathrm{n},\mathrm{m})$;

questo è un vincolo per la *KB* $\mathcal{K}$: ogni *consistency check* controlla che sia rispettato.

3. Ragionamento sulle membership function dei concetti concreti.

Se A, B sono concetti concreti, con membership function μ_A, μ_B allora

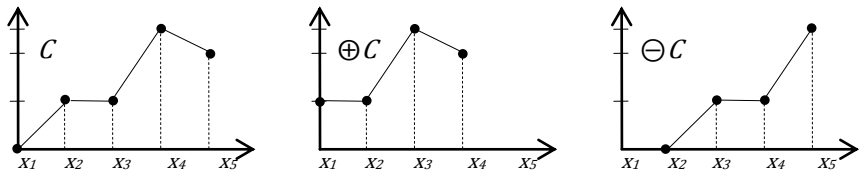

Figura 7-1 C, $\oplus C$, $\ominus C$

$$m = \langle \mu_A \subseteq \mu_B \rangle; \; n = max(n,m);$$

dati due concetti concreti A e B, con membership function esplicita, il ragionamento di sussunzione è una questione geometrico – analitica su un dominio concreto e fornisce il grado massimo al di sotto del quale la membership function di A è contenuta nella membership function di B.

4. Ragionamento sull'*ABox* (+ *TBox*).

Per ogni $x \in U$, determinare $\langle x:A, p\rangle$, $\langle x:B, q\rangle$: le asserzioni contenute nell'*ABox* stabiliscono quali individui dell'Universo U o dominio appartengano ai concetti presenti e con quale grado. I concetti sono insiemi contenenti gli individui dell'Universo, ognuno con un grado compreso tra 0 e 1. Se l'individuo x appartiene al concetto A con grado p e appartiene al concetto B con grado q, scriveremo

$$\langle x:A, p\rangle, \langle x:B, q\rangle.$$

I valori effettivi p e q sono assegnati dall'interpretazione:

$$p = A(x)^{\mathcal{I}}, \; q = B(x)^{\mathcal{I}}.$$

Utilizziamo quindi il metodo *GD,* che ci fornisce la cardinalità fuzzy relativa tra i due insiemi, e allora il grado di sussunzione m tra i due concetti è dato da

$$m = GD(B/A) = \sum_{\substack{\alpha_i \in \Lambda(B/A) \\ (A^I)_{\alpha_i} \subseteq (B^I)_{\alpha_i}}} (\alpha_i - \alpha_{i+1}) \qquad 7\text{-}7$$

dove gli α_i sono i livelli-α degli insiemi, ordinati dall'alto al basso: $\alpha_i \in \Lambda(B/A) = \Lambda(A) \cup \Lambda(B \cap A) = \{\alpha_1, .., \alpha_p\}$ e $\alpha_i > \alpha_{i+1}$ per ogni $i \in \{1, .., p\}$ e $\alpha_0 = 1$, $\alpha_{p+1} = 0$.

$$n = max(n,m);$$

5. restituire n.

Non affronto in questa sede le questioni di complessità.

7.3 Allegati: codice e documentazione

Qualche brano del codice realizzato.

file TBox.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="TBox" type="TBoxType"/>
  <xs:complexType name="TBoxType">
    <xs:sequence>
      <xs:element ref="simpleConcepts" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref="quantifiers" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="roles" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="query" type="QueryType"/>
```

```
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="QueryType">
      <xs:attribute name="quantifier" type="xs:string" />
      <xs:attribute name="role" type="xs:string" />
      <xs:attribute name="concept" type="xs:string" />
      <xs:attribute name="operator" type="xs:string" />
      <xs:attribute name="concept2" type="xs:string" />
      <xs:attribute name="isRoleQuery" type="xs:boolean" />
      <xs:attribute name="tempQuantifier" type="xs:string" />
  </xs:complexType>
  <xs:element name="simpleConcepts" type="SimpleConcept"/>
  <xs:element name="quantifiers" type="QuantifierType"/>
  <xs:element name="roles" type="RoleType"/>
  <xs:complexType name="SimpleConcept">
    <xs:sequence>
      <xs:element name="trapezium" type="TrapeziumType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="QuantifierType">
    <xs:sequence>
      <xs:element name="trapezium" type="TrapeziumType"/>
    </xs:sequence>
    <xs:attribute name="relative" type="xs:boolean"/>
  </xs:complexType>
  <xs:complexType name="TrapeziumType">
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="meaning" type="xs:string"/>
    <xs:attribute name="minVal" type="xs:double"/>
    <xs:attribute name="maxVal" type="xs:double"/>
    <xs:attribute name="x_a" type="xs:double"/>
    <xs:attribute name="x_b" type="xs:double"/>
    <xs:attribute name="x_c" type="xs:double"/>
    <xs:attribute name="x_d" type="xs:double"/>
  </xs:complexType>
  <xs:complexType name="RoleType">
    <xs:sequence>
      <xs:element name="domain" type="SimpleConcept"/>
      <xs:element name="range" type="SimpleConcept"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="functional" type="xs:boolean"/>
  </xs:complexType>
  <xs:complexType name="OperatorType">
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

file (Most)SubsNext(Warm).xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TBox>
    <simpleConcepts>
        <trapezium x_d="35.0" x_c="15.0" x_b="0.0" x_a="0.0" name="Cold"
minVal="-INF" meaning="LessThan(15)" maxVal="60.0"/>
    </simpleConcepts>
```

```
    <simpleConcepts>
        <trapezium x_d="60.0" x_c="60.0" x_b="35.0" x_a="0.0" name="Warm"
minVal="-INF" meaning="Between(35)and(60)" maxVal="60.0"/>
    </simpleConcepts>
    <quantifiers relative="true">
        <trapezium x_d="1.0" x_c="1.0" x_b="0.6" x_a="0.4" name="Most"
minVal="0.0" meaning="Between(0,6)and(1)" maxVal="1.0"/>
    </quantifiers>
    <query quantifier="Most" operator="SubsNext" isRoleQuery="false"
concept2="Warm" concept="Warm"/>
</TBox>
```

file reasoner.KnowledgeBase.java, metodo `xmlSave()`

```
      /**
       * Executes a saving in xml format for this KnowledgeBase's TBox
       * @param s the file destination
       * @throws JAXBException
       * @throws IOException
       */
      public synchronized void xmlSave(File s) throws JAXBException,
IOException{
            ObjectFactory factory = new ObjectFactory();
            TBoxType tbox = factory.createTBoxType();
            Set<String> keys;
            Iterator<String> it;
            keys = this.getConcrConcepts().keySet();
            it = keys.iterator();
            while(it.hasNext()){
                  ConcreteConcept c = (ConcreteConcept)
this.getConcrConcepts().get(it.next());
                  SimpleConcept sc = factory.createSimpleConcept();

                  TrapeziumType t =factory.createTrapeziumType();
                  t.setName(c.getName());
                  t.setMeaning(c.getMeaning());
                  t.setMinVal(c.getMinval());
                  t.setMaxVal(c.getMaxval());
                  t.setXA(c.geta());
                  t.setXB(c.getb());
                  t.setXC(c.getc());
                  t.setXD(c.getd());
                  sc.setTrapezium(t);
                  tbox.getSimpleConcepts().add(sc);
            }
            keys = this.getQuantifiers().keySet();
            it = keys.iterator();
            while(it.hasNext()){
                  Quantifier q = (Quantifier)
this.getQuantifiers().get(it.next());
                  QuantifierType qt = factory.createQuantifierType();
                  TrapeziumType t =factory.createTrapeziumType();
                  t.setName(q.getName());
                  t.setMeaning(q.getMeaning());
                  t.setMinVal(q.getMinval());
                  t.setMaxVal(q.getMaxval());
                  t.setXA(q.geta());
                  t.setXB(q.getb());
```

```
                t.setXC(q.getc());
                t.setXD(q.getd());
                qt.setTrapezium(t);
                qt.setRelative(q.isRelative());
                tbox.getQuantifiers().add(qt);
          }
          keys = this.getRoles().keySet();
          it = keys.iterator();
          while(it.hasNext()){
                Role r = (Role) this.getRoles().get(it.next());
                RoleType rt = factory.createRoleType();
                rt.setName(r.getName());
                tbox.getRoles().add(rt);
          }
          QueryType qt = factory.createQueryType();
          Query qry = this.getQuery();
          if(qry!=null){
                if (qry instanceof reasoner.RoleQuery){
                      qt.setIsRoleQuery(true);
                      if(null!=((RoleQuery)qry).getTempq())

    qt.setTempQuantifier(((RoleQuery)qry).getTempq().getName());
                      if(null!=((RoleQuery)qry).getQ())

    qt.setQuantifier(((RoleQuery)qry).getQ().getName());
                      qt.setRole(((RoleQuery)qry).getR().getName());

    qt.setConcept(((RoleQuery)qry).getTc().getName());
                }else if(qry instanceof reasoner.OpQuery){
                      qt.setIsRoleQuery(false);
                      if(null!=((OpQuery)qry).getQ())

    qt.setQuantifier(((OpQuery)qry).getQ().getName());
                      qt.setConcept(((OpQuery)qry).getTc1().getName());
                      qt.setOperator(((OpQuery)qry).getOperator());

    qt.setConcept2(((OpQuery)qry).getTc2().getName());
                }
                tbox.setQuery(qt);
          }
          JAXBElement<TBoxType> tboxEl =
(JAXBElement<TBoxType>)factory.createTBox(tbox);
          JAXBContext ctx = JAXBContext.newInstance("tbox");
          Marshaller m = ctx.createMarshaller();
    m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
      FileOutputStream out = new FileOutputStream(s);
      m.marshal(tboxEl, out);
      out.flush();
      out.close();
    }
```

file reasoner.Concept.java, metodo `subsNext()`

```
/**
       * Creates the new Concept corresponding to the logical expression
SubsNext(C), where
       * C is the present Concept
       * @param fuzzy a boolean, set to true if fuzzy subsumption is
needed
```

```
       * @return the SubsNext(C) Concept
       */
      public Concept subsNext(boolean fuzzy){
            Concept next, inter, subsNext;
            HashMap<String, ULineEvent> uLines =
getKb().getULineEvents();
            HashSet<String> ulNames = new HashSet<String>();
            ULineEvent ul, ul1;
            Individual xlast;
            ULineInd xo;
            Double newVal = 0.0;
            subsNext = new Concept("SubsNext"+ this.getName(), getKb());
            ulNames.addAll((Set<String>)uLines.keySet());//
            for (String ulName: ulNames){
                  ul = (ULineEvent) uLines.get(ulName);
                  xlast = ul.getLast();
                  xo = ul.getMyULine();
                  next= this.getNextConcept(xlast);
                  ul1 = ul.getWithoutLast();
                  inter = this.intersection(ul1);
                  newVal = inter.subsumed(next, fuzzy);
                  getKb().getULineEvents().remove(ul1.getName());
                  subsNext.add(xo, newVal);
            }
            return subsNext;
      }
```

file reasoner.Concept.java, metodo `createTendencyConcepts()`

```
/**
       * Creates two new Concepts corresponding to the logical
expressions SubsNext(C) and
       * SubsPrev(C), where C is the present Concept, and consequently
creates the three Tendency
       * Concepts Incr(C), Decr(C) and Const(C).
       * @param fuzzy a boolean, set to true if fuzzy subsumption is
needed
       */
      public void createTendencyConcepts(boolean fuzzy) {
            ArrayList<Concept> ar = this.subsNextPrev(fuzzy);
            HashMap<String, ULineEvent> uls = getKb().getULineEvents();
            Set<String> ulNames = uls.keySet();
            Concept next, prev;
            next = ar.get(0);
            prev = ar.get(1);
            Concept incr, decr, constant;
            incr = new Concept("Incr(" + this.getName() + ")", getKb());
            decr = new Concept("Decr(" + this.getName() + ")", getKb());
            constant = new Concept("Const(" + this.getName()+ ")",
getKb());
            ULineEvent ul;
            ULineInd xo;
            double nextval, prevval;
            for (String name : ulNames){
                  ul = uls.get(name);
                  xo = ul.getMyULine();
                  nextval = next.get(xo);
                  prevval = prev.get(xo);
                  if (nextval > prevval)
```

```
                        incr.add(xo, nextval - prevval);
                  else if (nextval < prevval)
                        decr.add(xo, prevval - nextval);
                  else if (nextval == prevval)
                        constant.add(xo, nextval);
            }
      }
```

file currentState.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="CurrentState" type="CurrentStateType"/>
      <xs:complexType name="CurrentStateType">
            <xs:sequence>
                  <xs:element ref="nodes"  minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string"/>
                <xs:attribute name="time" type="xs:dateTime"/>
        </xs:complexType>
      <xs:element name="nodes" type="NodeType"/>
      <xs:complexType name="NodeType">
            <xs:attribute name="coord_x" type="xs:double"/>
            <xs:attribute name="coord_y" type="xs:double"/>
            <xs:attribute name="name" type="xs:string"/>
      </xs:complexType>
</xs:schema>
```

file CurrentState.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CurrentState time="2008-01-01T18:59:15.468+01:00">
    <nodes name="Pray" coord_y="673.9360971440653"
coord_x="169.9855068521619"/>
    <nodes name="One" coord_y="629.317685273028"
coord_x="207.58640595066325"/>
    <nodes name="Two" coord_y="712.7127745365442"
coord_x="195.1318094163467"/>
    <nodes name="Three" coord_y="623.0312499058969"
coord_x="129.29754309020962"/>
</CurrentState>
```

esempio di monitoring in tempo reale:

frammento di file log.txt d'inizio esecuzione per la query *(Most)Incr(CloseDistance)*:

```
 One:  1
 Two:  0,71493
 Three:  0,964089
Query evaluation:
  SubsNext(CloseDistance): {<one, 1>, <three, 1>, <two, 1>}
  SubsPrev(CloseDistance): {<one, 1>, <three, 1>, <two, 1>}
  Incr(CloseDistance): {}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 1>, <three, 1>, <two, 1>}
  linear interpolation: {Three: angCoeff = 0; One: angCoeff = 0; Two:
angCoeff = 0}
```

```
      (Most)Incr(CloseDistance): 0
**********************************
 One:  1
 Two:  0,73014
 Three:  0,987057
Query evaluation:
  SubsNext(CloseDistance): {<one, 1>, <three, 1>, <two, 1>}
  SubsPrev(CloseDistance): {<one, 1>, <three, 0,977>, <two, 0,985>}
  Incr(CloseDistance): {<three, 0,023>, <two, 0,015>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 1>}
  linear interpolation: {Three: angCoeff = 0,002; One: angCoeff = -0;
Two: angCoeff = 0,002}

      (Most)Incr(CloseDistance): 0,015209
**********************************
 One:  1
 Two:  0,755977
 Three:  1
Query evaluation:
  SubsNext(CloseDistance): {<one, 1>, <three, 1>, <two, 1>}
  SubsPrev(CloseDistance): {<one, 1>, <three, 0,964>, <two, 0,959>}
  Incr(CloseDistance): {<three, 0,036>, <two, 0,041>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 1>}
  linear interpolation: {Three: angCoeff = 0,003; One: angCoeff = -0;
Two: angCoeff = 0,003}

      (Most)Incr(CloseDistance): 0,035911
**********************************
 One:  1
 Two:  0,769613
 Three:  1
Query evaluation:
  SubsNext(CloseDistance): {<one, 1>, <three, 1>, <two, 1>}
  SubsPrev(CloseDistance): {<one, 1>, <three, 0,964>, <two, 0,945>}
  Incr(CloseDistance): {<three, 0,036>, <two, 0,055>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 1>}
  linear interpolation: {Three: angCoeff = 0,002; One: angCoeff = -0;
Two: angCoeff = 0,004}

      (Most)Incr(CloseDistance): 0,035911
**********************************
 One:  1
 Two:  0,831034
 Three:  1
Query evaluation:
  SubsNext(CloseDistance): {<one, 1>, <three, 1>, <two, 1>}
  SubsPrev(CloseDistance): {<one, 1>, <three, 0,964>, <two, 0,884>}
  Incr(CloseDistance): {<three, 0,036>, <two, 0,116>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 1>}
  linear interpolation: {Three: angCoeff = 0,002; One: angCoeff = -0;
Two: angCoeff = 0,007}

      (Most)Incr(CloseDistance): 0,035911
**********************************
 One:  1
```

```
 Two:  0,861751
 Three:  1
Query evaluation:
  SubsNext(CloseDistance): {<one, 1>, <three, 1>, <two, 1>}
  SubsPrev(CloseDistance): {<one, 1>, <three, 0,964>, <two, 0,853>}
  Incr(CloseDistance): {<three, 0,036>, <two, 0,147>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 1>}
  linear interpolation: {Three: angCoeff = 0,002; One: angCoeff = -0;
Two: angCoeff = 0,008}

      (Most)Incr(CloseDistance): 0,035911
**********************************
 One:  1
 Two:  0,889508
 Three:  0,985346
Query evaluation:
  SubsNext(CloseDistance): {<one, 1>, <three, 0,985>, <two, 1>}
  SubsPrev(CloseDistance): {<one, 1>, <three, 0,964>, <two, 0,825>}
  Incr(CloseDistance): {<three, 0,021>, <two, 0,175>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 1>}
  linear interpolation: {Three: angCoeff = 0,001; One: angCoeff = -0;
Two: angCoeff = 0,01}

      (Most)Incr(CloseDistance): 0,021257
**********************************
 One:  1
 Two:  0,957058
 Three:  0,995066
Query evaluation:
  SubsNext(CloseDistance): {<one, 1>, <three, 0,985>, <two, 1>}
  SubsPrev(CloseDistance): {<one, 1>, <three, 0,964>, <two, 0,758>}
  Incr(CloseDistance): {<three, 0,021>, <two, 0,242>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 1>}
  linear interpolation: {Three: angCoeff = 0,001; One: angCoeff = -0;
Two: angCoeff = 0,014}

      (Most)Incr(CloseDistance): 0,021257
**********************************
 One:  1
 Two:  0,955582
 Three:  0,994157
Query evaluation:
  SubsNext(CloseDistance): {<one, 1>, <three, 0,985>, <two, 0,999>}
  SubsPrev(CloseDistance): {<one, 1>, <three, 0,964>, <two, 0,758>}
  Incr(CloseDistance): {<three, 0,021>, <two, 0,241>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 1>}
  linear interpolation: {Three: angCoeff = 0; One: angCoeff = -0; Two:
angCoeff = 0,014}

      (Most)Incr(CloseDistance): 0,021257
**********************************
 One:  1
 Two:  0,989275
 Three:  0,997732
Query evaluation:
```

```
  SubsNext(CloseDistance): {<one, 0,987>, <three, 0,985>, <two, 0,988>}
  SubsPrev(CloseDistance): {<one, 0,987>, <three, 0,964>, <two, 0,715>}
  Incr(CloseDistance): {<three, 0,021>, <two, 0,273>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 0,987>}
  linear interpolation: {Three: angCoeff = 0,001; One: angCoeff = -0;
Two: angCoeff = 0,016}

      (Most)Incr(CloseDistance): 0,021257
**********************************
 One:  1
 Two:  1
 Three:  1
Query evaluation:
  SubsNext(CloseDistance): {<one, 0,987>, <three, 0,985>, <two, 0,988>}
  SubsPrev(CloseDistance): {<one, 0,987>, <three, 0,965>, <two, 0,716>}
  Incr(CloseDistance): {<three, 0,02>, <two, 0,272>}
  Decr(CloseDistance): {}
  Const(CloseDistance): {<one, 0,987>}
  linear interpolation: {Three: angCoeff = 0,001; One: angCoeff = -0;
Two: angCoeff = 0,016}

      (Most)Incr(CloseDistance): 0,020244
**********************************
```

Ringraziamenti

Desidero ringraziare le molte persone che hanno contribuito alla realizzazione di questa tesi, che è nata e si è sviluppata in condizioni oggettivamente ristrette.

Il Prof. Marco Colombetti, che è stato assai più che un relatore: mi ha concesso di condividere il suo tempo in lunghe e interessanti conversazioni, stimolandomi nello studio e nella ricerca nei settori appassionanti dell'Ingegneria della Conoscenza e dell'Intelligenza Artificiale.

Il Prof. Marco Locatelli, che mi ha sostenuto totalmente in tutto il mio percorso di studi, dandomi un sostegno e supporto essenziale e continuo: senza di lui, questa tesi non avrebbe mai visto la luce. Egli è uno dei miei più cari amici.

La Dott.ssa M. Grazia Moi, che mi ha concesso le preziose opportunità di frequentazione del Politecnico e d'incontro con i professori.

L'Ing. Davide Eynard, che è stato un consigliere generoso e stimolante.

L'Ing. David Laniado, che mi ha offerto la sua competenza e simpatia.

Il Prof. Arturo Locatelli, che si è spesso attivato in silenzio perché il percorso si compisse.

M. Crista, sempre presente, nelle gioie e nei dolori, che ha arricchito di emotività questa tesi.

Mons. Guido Todeschini, che mi ha accompagnato nella riscoperta spirituale, donando nuovo significato alla mia vita.

I miei genitori, naturalmente: mio padre critico assiduo, mia madre più benevola, con dedizione straordinaria. Entrambi avrebbero meritato maggior attenzione.

La Sig.ra Enrica Spreafico, che ha contribuito al supporto logistico e quindi alla mia necessaria serenità.

La Direzione e il Comando, nei vari gradi, della C.R. Opera, che mi hanno permesso di portare a termine i miei studi, attivandosi a superare, ove possibile, le naturali difficoltà del luogo.

E tanti altri che non nomino, ma non dimentico.

Indice delle figure

Indice delle tabelle

Bibliografia

1. **A. Artale, and E. Franconi.** A surview of temporal extensions of description logics. *Annals of Mathemthics and Artificial Intelligence.* 2001, 30(1-4).

2. *The semantic web.* **T. Berners-Lee, J. Hendler, and O. Lassila.** 2001, The Scientific American, p. 284(5):34-43.

3. **Aristotele.** *Metafisica, libri V, VII, VII, IX.*

4. **Zadeh, L.A.** Fuzzy sets. *Information and Control.* 1965, 8(3):338-353.

5. **Straccia, Umberto.** *Towards a Fuzzy Description Logic for the semantic Web.* Pisa, Italy : Istituto di Scienza e Tecnologie dell'Informazione, CNR, 2006.

6. *A definition of nonprobabilistic entropy in the setting of fuzzy sets theory.* **De Luca A., and Termini S.** 20, 1972, p. 301-312.

7. **M. Delgado, M.J. Martin-Bautista, D. Sanchez, and M.A. Vila.** A probabilistic definition of a nonconvex fuzzy cardinality. *Fuzzy Sets and Systems.* 2002, 126(2):41-54.

8. **Ralescu, D.** Cardinality, quantifiers and the aggregazion of fuzzy criteria. *Fuzzy Sets and Systems.* 1995, Vol. 69, p. 355-365.

9. **D. Dubois, H. Prade.** Fuzzy cardinality and the modeling of imprecise quantification. *FUzzy Sets and Systems.* 1985, Vol. 16, p. 199-230.

10. *On the best scalar approximation of cardinality of a fuzzy set.* **Wygralak, M.** 1997, Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 5 (6), p. 681-687.

11. **Zadeh, L. A.** A computational approach to fuzzy quantifiers in natural languages. *Comput. Math. Appl.* 1983, 9 (1), p. 149-184.

12. —. A theory of approximate reasoning. *Machine Intell.* 1979, 9, p. 149-194.

13. *Triangular norm-based generalized cardinals for fuzzy sets.* **Wygralak, M.** 1999. Proceeding of the 7th Zittan Fuzzy Colloquium. p. 234-239.

14. **M. Delgado, D. Sanchez, and M.A. Vila.** Fuzzy cardinality based evaluation of quantified sentences. *International Journal of Approximate Reasoning.* 2000, 23:23-66.

15. **Emerson, E. A.** Temporal and Modal Logic. *Handbook of Theoretical Computer Science.* s.l. : North-Holland Pub. Co., 1995.

16. **J. Y. Halpern, and Y. Shoham.** A propositional modal logic of time intervals. *Journal of ACM.* 1991, 39(4):935-962.

17. **P. T. Devanbu, and D. J. Litman.** Taxonomic plan reasoning. *Artificial Intelligence.* 1996, 84:1-35.

18. **M. Finger, and D. Gabbay.** Adding a temporal dimension to a logic system. *Journal of Language and Information.* 1993, 1:203-233.

19. *Foundation of spatiotemporal reasoning with description logics.* **V. Haarslev, C. Lutz, and R. Moeller.** Trento, Italy : s.n., 1998, Proc. of the 6th Intl Conference on Knowledge Representation and Reasoning, KR'98, p. 112-123.

20. **A. Artale, and E. Franconi.** Temporal description logics. [aut. libro] P. vanBeek, M, Boddy, M. Fisher, D. Gabbay, A. Galton, and R. Morris L. Vila. *Handbook of Time and Temporal Reasoning in Artificial Intelligence.* s.l. : MIT Press, 1999.

21. *The DLRUS Temporal Description Logic.* **Artale A., Franconi E., Mosurovic M.,Wolter F., Zakharyaschev M.** 2001.

22. **K. Ben Lamine, and F. Kabanza.** Using fuzzy temporal logic for monitoring behavior- based mobile robots.

23. *Generalizing quantification in fuzzy description logics.* **D. Sànchez, and G.B Tettamanzi.** Dortmund : s.n., 2004. Proc. of the 8th Fuzzy Days.

Sommario