

# Un formalismo basato sulla Frame Logic per il ragionamento spaziale. Un'applicazione al problema della visualizzazione di relazioni spaziali qualitative

Licenziato Luca\*, Mele Francesco\*\*

\*Università di Napoli Federico II

Via Cinzia Napoli, 80126 Napoli [licenzia@studenti.unina.it](mailto:licenzia@studenti.unina.it)

\*\* Istituto di Cibernetica Consiglio Nazionale delle Ricerche,

Via dei Campi Flegrei 34 Pozzuoli (NA) [f.mele@cib.na.cnr.it](mailto:f.mele@cib.na.cnr.it)

## Abstract

Il principale obiettivo del presente lavoro è stato quello di costruire un formalismo basato su una rappresentazione a frame, che catturasse tutti gli aspetti significativi (anche complessi) fra le parti e la totalità di un oggetto fisico composto, e le relazioni spaziali qualitative delle parti appartenenti alla totalità.

Il lavoro è stato svolto attraverso una prima fase di rigorosa scelta, della rappresentazione di base che ci ha portato al formalismo proposto, il quale è stato definito rispettando una serie di requisiti guida esistenti nel settore, tra questi, quello di adottare un approccio esplicito per la rappresentazione delle totalità. Inoltre l'implementazione della base di conoscenza è stata realizzata utilizzando un paradigma basato su prototipi – paradigma che ha come nozioni di base il prototipo e la delega, diversamente dal paradigma basato su classi, che ha la classe e l'ereditarietà.

Una seconda fase è stata dedicata alla costruzione di una rappresentazione, e di un'assiomatica, per il ragionamento spaziale qualitativo, integrata nei requisiti guida. Infine, una terza fase è stata dedicata alla costruzione di un ragionatore, sviluppato in Frame Logic, che fosse in grado, utilizzando l'apparato inferenziale presente nell'assiomatica, di eseguire il completamento e il rendering di oggetti 3D.

## 1 Introduzione

Il ragionamento spaziale è stato, fino ad ora, un argomento che si è sviluppato all'interno di diverse discipline come la Robotica, la Fisica Qualitativa e l'Analisi delle immagini. Recentemente è diventato un argomento di studio con una precisa missione e connotazione. La missione è quella di definire formalismi, al fine di potere *catturare* le molteplici tipologie di relazioni e inferenze spaziali esistenti, in diversi domini conoscitivi (quasi ogni dominio possiede una specificità propria) al fine di potere *effettuare* ragionamenti spaziali intorno ad oggetti collocati nello spazio. Un caso particolare costituisce lo studio del ragionamento spaziale qualitativo umano. Modello ritenuto interessante perché si crede possa essere adoperato in diverse tipologie di applicazioni che coinvolgono il ragionamento spaziale.

L'ambito di collocazione del ragionamento spaziale, invece, è quello delle ontologie formali [2], settore che ha messo a disposizione molti strumenti per affrontare le varietà di problemi esistenti nel settore in esame.

Un'attiva ricerca specifica, nel settore del ragionamento spaziale, riguarda lo studio delle relazioni fra una totalità e le sue parti [1]. In questo studio le relazioni spaziali (*sta\_sotto\_a*, *sta\_a\_destra\_di*, ect) sono usate per descrivere la struttura di tali oggetti composti.

Riguardo all'approccio metodologico adottato in questo lavoro, faremo riferimento ai requisiti di base presentati in [1]. Nel citato lavoro, si propone un insieme di requisiti (minimi a parere degli autori) per costruire un modello idoneo "a catturare l'ontologica natura delle parti e delle totalità" di oggetti fisici composti:

- 1- Adottare un approccio esplicito per la definizione delle totalità
- 2- Eseguire una chiara distinzione fra le parti ed altri attributi di una totalità
- 3- Costruire in maniera built-in la relazione di transitività fra parti

- 4- Possibilità di riferirsi alle parti mediante generici nomi
- 5- Capacità di esprimere relazioni fra parti e totalità
- 6- Capacità di esprimere relazioni fra le parti di una totalità

Una rappresentazione di oggetto composto è implicita quando nella definizione della totalità non sono presenti (esplicitamente) le parti che compongono la totalità. In questi tipi di rappresentazioni sono le parti che *hanno conoscenza* a quali totalità appartengono, conoscenza espressa localmente nella definizione delle parti stesse.

Contrariamente, una rappresentazione è esplicita se nella definizione della totalità, che rappresenta l'oggetto composto, sono presenti esplicitamente le connessioni con le parti che la compongono, in altre parole, quando la totalità *conosce* le sue parti. Un esempio dei due tipi di rappresentazione è il seguente:

#### Approccio implicito

colonna1 : stile : dorico

capitello1 : descrizione non strutturale: xxxx  
 : tipoforma : cubo  
 : altezza : 1.0 m  
 : parte\_di: colonna1

fusto1 : descrizione non strutturale : xxxx  
 : tipoforma : cilindro  
 : altezza : 4.0 m  
 : raggio\_base : 5  
 : parte\_di: colonna1

#### Approccio esplicito

colonna1 : stile : dorico  
 : formata\_da : capitello1, fusto1

capitello1 : descrizione non strutturale: xxxx  
 : tipoforma : cubo  
 : altezza : 1.0 m

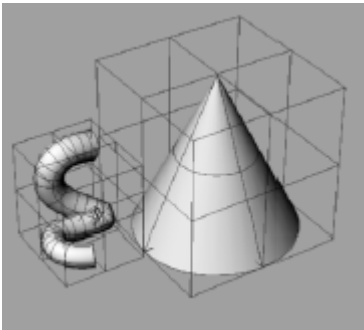
fusto1 : descrizione non strutturale : xxxx  
 : tipoforma : cilindro  
 : altezza : 4.0 m  
 : raggio\_base : 5

Una rappresentazione esplicita comporta alcuni vantaggi.

In primo luogo *espressività* e *dichiaratività*: nella definizione di una totalità le parti sono esplicitamente dichiarate, contrariamente a quello che accade in una rappresentazione implicita, dove, per avere il *quadro* della completa struttura, bisogna *andare a ricercare* (in maniera procedurale), nelle definizioni delle parti, quelle che appartengono alla totalità che si sta esaminando.

In secondo luogo il *riuso*: è semplice convincersi che, se si rappresentano le parti in maniera da non contenere l'attributo di appartenenza alla totalità, tali definizioni non dipendono dal contesto di appartenenza (dalle totalità). Per tale ragione risulta che una definizione di parte, così strutturata, può essere utilizzata da più definizioni di totalità.

Se ad un approccio esplicito si aggiunge la possibilità di riferirsi alle parti mediante generici nomi (punto 4 della precedente lista), si conferisce ad un formalismo proprietà di composizionalità: avere la possibilità di costruire nuove totalità, in maniera incrementale, ossia, comporre nuove aggregazioni di oggetti senza dover modificare le definizioni di altre precedentemente definite.



Come faremo vedere nel formalismo di questo lavoro, abbiamo scelto un approccio esplicito per la rappresentazione delle totalità. Nell'approccio proposto, una totalità è definita mediante l'insieme delle sue parti (come nel caso di definizione esplicita dell'esempio riportato). In aggiunta, nella nostra proposta, una totalità è definita a) mediante l'insieme delle relazioni della totalità con le sue parti b) mediante l'insieme delle relazioni fra le parti stesse.

I punti a) e b) ci hanno permesso di raggiungere i requisiti guida 5) e 6) della lista. Per il punto 5) "la capacità di esprimere relazioni fra parti e totalità" abbiamo utilizzato una struttura geometrica di riferimento: il BoundingBox<sup>1</sup>. L'oggetto-totalità è la composizione dei suoi oggetti-parti, ed il BoundingBox rappresenta il risultato di questa

<sup>1</sup> Il BoundingBox è il parallelepipedo, con i lati paralleli agli assi cartesiani, che contiene un oggetto tridimensionale nello spazio, nel nostro caso contiene la totalità.

composizione. Nello specifico delle relazioni spaziali, invece, una parte si *relaziona male* con la sua totalità, per il semplice fatto che, da un punto di vista strutturale-geometrico, la totalità stessa possiede pochi attributi intrinseci descrittivi. Scegliendo come riferimento il BoundingBox abbiamo potuto esprimere, nel formalismo proposto, tutte le relazioni spaziali di base fra gli oggetti, definendo una teoria di composizione tra le parti per rappresentare le totalità.

A riguardo del punto 6), invece, abbiamo definito un formalismo, che utilizza un insieme esteso di relazioni spaziali qualitative, *vincolate a rispettare* un insieme di assiomi e un insieme di proprietà (si veda avanti). Il formalismo è un sottoinsieme di Flora2[6] un'implementazione della Frame Logic[15]. Le regole, le estensioni, i vincoli ed il ragionatore, che proponiamo in questo lavoro, sono interamente definiti in Flora2, anche se, per motivi che riporteremo in avanti, abbiamo implementato un meccanismo prototype-based differente da quello standard, class-based, di Flora2. Gli elementi di base del linguaggio Flora2 sono i seguenti:

(per la definizione delle classi e le relazioni tassonomiche fra classi)

```
x::z                x è una sottoclasse z;
x[s =>y]            s è un attributo di tipo y della classe x;
y[s=>> {t1,t2, ..}] s è uno slot di y a valori multipli di tipo t1,t2, ..;
```

(per la definizione delle istanze)

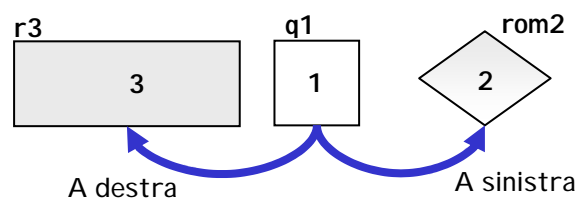
```
a:x                a è istanza di y;
idx: x[s->vx]      vx è il valore dell'attributo s della classe x;
idx: x[s->>{v1,v2, ..}] v1, v2, .. sono i valori dell'attributo s della classe x.
```

(idx è l'identificatore logico dell'oggetto - conosciuto nella programmazione ad oggetti come Oid)

Riportiamo un esempio:

```
/* Tassonomia e definizione delle classi */
parallelogrammi::figure_piane.
rettangoli::parallelogrammi.
rombi::parallelogrammi.
quadrati::rettangoli.
triangoli::figure_piane.
figure_piane[numero_figura=>integer,
              colore=>string,
              a_sinistra=>figure_piane,
              a_destra=>figure_piane].
rombi[diagonale1=> integer, diagonale2=> integer].
rettangoli[base=> integer, altezza=> integer]
```

```
/* Istanze */
q1:quadrati[base->3, colore->bianco,
            numero_figura->1, a_destra->r3].
r3:rettangoli[altezza->3, base->7,
              colore->grigio, numero_figura->3].
rom2:rombi[colore->biancogrigio,numero_figura->2,
            diagonale1->3,diagonale2->5,
            a_destra->q1].
```



```
/* Assiomi */
Y[a_sinistra->X]:-X[a_destra->Y]. (inverse slot)
X[a_destra->Y]:-Y[a_sinistra->X].
X[a_sinistra->Z]:-X[a_sinistra->Y],Y[a_sinistra->Z].
X[a_destra->Z]:-X[a_destra->Y],Y[a_destra->Z]
quadrati[altezza->X, base->X].
```

```
/* Esempi di query */
?- I:romb1[colore->Y].
?- L:romb1[a_sinistra->X].
?- I:C[A->V].
```

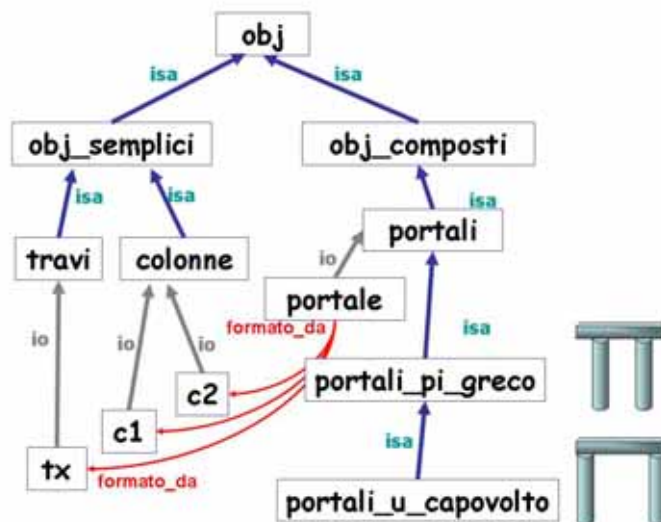
L'approccio da noi utilizzato, a riguardo del paradigma computazionale, è stato quello di adottare una metodologia che si fonda su un paradigma denominato in letteratura, prototype – based[16]. Esistono due modi di rappresentare i concetti generali: tramite insiemi e tramite prototipi. Un primo modo è quello di costruire il concetto di classe, un'entità che astrae le caratteristiche comuni ad un insieme di oggetti i quali sono istanze di questa classe. Caratteristiche aggiuntive possono essere contenute nelle sottoclassi, e la condivisione tra

classe e sottoclasse avviene mediante l'ereditarietà. Un secondo modo è quello basato sui prototipi e sulla delega[16]. La delega rappresenta l'operazione di affidare ad un altro oggetto un compito, qualora un oggetto non è in grado di soddisfare una particolare richiesta. È possibile mediante un tale approccio definire prototipi rappresentanti "famiglia" di oggetti, dove ogni oggetto condivide con il prototipo tutte le informazioni comuni, e nello stesso tempo ha immagazzinato nella propria struttura lo che lo caratterizzano e lo differenziano dal prototipo.

Nell'approccio di rappresentazione che proponiamo, al fine di costruire tassonomie di concetti, utilizziamo gli standard concetti di classe, sottoclasse ed istanza dei linguaggi orientati agli oggetti e della Frame Logic. Ma, diversamente a quanto accade in tali approcci, arricchiamo le definizioni delle classi con prototipi – strutture che sono utilizzate per effettuare eventuali completamenti che si rendono necessari per le istanziazioni degli oggetti.

## 2 Formalismo e paradigma

In questo paragrafo presenteremo un formalismo per il ragionamento spaziale qualitativo, mostrando, di volta in volta, come sia conforme alle linee guida, elencate nell'introduzione, presentate in [1]. Il nostro formalismo è orientato al ragionamento al fine di costruire una scena, a partire da relazioni spaziali anche non espressamente date. Ciò è stato fatto costruendo un apparato deduttivo (assiomatico).



Uno dei principali obiettivi che ci siamo posti per la costruzione del formalismo, è stato quello di voler definire un linguaggio che permettesse di rappresentare classi di oggetti fisici quanto più dettagliate e specifiche, e nello stesso tempo, che permettesse di *catturare* eventuali sfumature, fra le classi di una tassonomia, adottando un graduale meccanismo di raffinamento. La sintassi e la semantica del formalismo proposto ricalcano, ovviamente, quella della Frame Logic, ed è presentato di seguito:

```
sottoclasse :: classe.          --definizione sottoclasse
istanza : classe.              --definizione di istanza/prototipo
istanza_ogg_semplice[ha_forma -> primitive, height->float, width->float, depth->float].
--definizione delle dimensioni del boundingbox e della primitive degli oggetti semplici nella
dichiarazione di istanza/prototipo
istanza_ogg_composto[formato_da->{{lista_oggetti}},relazioni_spaziali->{{lista_relazioni}}].
--definizione di oggetto composto, mediante F-molecola a due slot:
formato_da (contenente la lista degli oggetti semplici o composti),
relazioni_spaziali (contenente la lista delle relazioni spaziali sugli oggetti componenti)
istanza_relazione_spaziale[obj1->oggetto1, obj2->oggetto2].
--definizione di relazione spaziale, i due slot obj1 e obj2 rappresentano gli oggetti (semplici o
composti) ai quali si riferisce la relazione.
```

La BNF dell'intero formalismo è riportata in Appendice A. Per fornire una prima idea della rappresentazione adoperata, riportiamo un esempio nel quale è rappresentata una gerarchia di portali definiti all'interno della base di conoscenza, utilizzando oggetti semplici (*obj\_semplici*) e composti (*obj\_composti*)<sup>2</sup>.

```

/* Classi di oggetti semplici e composti */
obj_semplici::obj.
obj_composti::obj.
portali::obj_composti.
travi::obj_semplici.
colonne::obj_semplici.

/* Definizione prototipo per portali */
portale1:portali.
portale1[ formato_da ->> {tx, c1, c2} ].
/* Definizione prototipo per portale Pi-greco */
portali_pigreco::portali.
portale_pigreco1:portali_pigreco.
portale_pigreco1[ relazioni_spaziali->>{s1,s2,i1,i2},
                delega ->> {portale1} ].
/*Definizione prototipo per portale u capovolta*/
portali_u_cap::portali_pigreco.
portale_u_cap1:portali_u_cap.
portale_u_cap1[relazioni_spaziali ->>{adx1, asx1},
                delega ->> {portale_pigreco1}].

/* Oggetti semplici */
tx:travi.
c1:colonne.
c2:colonne.
travi [ha_forma -> parallepiedi].
colonne [ha_forma -> cilindri].

/* Relazioni spaziali la classe portali Pi-greco*/
s1:sopra[obj1->tx, obj2->c1].
s2:sopra[obj1->tx, obj2->c2].
i1:in_contatto[obj1->tx, obj2->c1].
i2:in_contatto[obj1->tx, obj2->c2].

/*Relazioni spaziali per prototipo portali_u_cap*/
asx1:allineato_asinistra[obj1->tx, obj2->c1].
adx1:allineato_adestra[obj1->tx, obj2->c2].

```

Come si può osservare, il formalismo permette di rappresentare le totalità (ad esempio *portale1*) in maniera tale da contenere esplicitamente, nella definizione, le parti di cui esse sono formate (*portali[ formato\_da ->> {tx, c1, c2}*), in altre parole il formalismo è esplicito.

Le parti stesse (*tx, c1, c2*), inoltre, possono essere sia oggetti semplici che a loro volta composti. Nell'esempio, ovviamente, la classe *portali* gioca il ruolo di classe astratta. Le classi concrete sono definite, nel formalismo proposto, definendo o ulteriori relazioni spaziali qualitative fra le parti o aggiungendo nuove parti. Per la classe *portali\_pigreco*, definita come sottoclasse della classe *portali*, le relazioni<sup>3</sup> *s1, s2, i1* e *i2* sono appunto relazione spaziali qualitative fra le parti *c1, c2* e *tx* della totalità *portali\_pigreco*. Nell'esempio, infine, a partire dalla classe *portali\_pigreco* è stata definita una sottoclasse *portale\_u\_cap*, dove le relazioni spaziali qualitative *adx1, asx1* conferiscono una particolare caratterizzazione alla classe (mediante il suo prototipo), ossia, di avere entrambe le colonne poste alle estremità della trave.

Vogliamo sottolineare che il formalismo rispetta le linee guida elencate in precedenza. Oltre ad adottare un approccio esplicito (punto 1), permette una chiara distinzione fra le parti e altri attributi di una totalità (punto 2). Il formalismo, inoltre, permette di riferirsi alle parti mediante generici nomi (punto 4), in modo tale da potere utilizzare una definizione di parte in più definizioni di totalità. A riguardo dei punti 5) e 6), come si può osservare anche dall'esempio riportato, il formalismo permette di esprimere relazioni fra parti e totalità (es. *formato\_da ->> {tx, c1, c2}*) e fra le parti di una totalità (es. *s1:sopra[obj1->tx, obj2->c1], i1:in\_contatto[obj1->tx, obj2->c1], asx1:allineato\_asinistra[obj1->tx, obj2->c1]*).

Come si può notare, inoltre, le istanze/prototipi di oggetti composti possiedono uno slot denominato *delega*, che contiene la lista di oggetti con i quali l'oggetto dato condivide

<sup>2</sup> La sintassi di tale formalismo è simile a quella proposta in [4][22].

<sup>3</sup> Nell'esempio abbiamo riportato una forma semplificata della relazione *sopra*, definita nel formalismo. In avanti riporteremo la forma completa di tale relazione.

informazioni (slot formato\_da e slot relazioni\_spaziali) secondo il meccanismo proprio del paradigma *prototype – based*.

Per supportare il paradigma *prototype – based*, dato che il modello computazionale è diverso da quello *class – based* in quanto non sfrutta l'ereditarietà, è stato necessario implementare un meccanismo di delega, per far sì che qualora un oggetto non fosse in grado di rispondere ad una certa richiesta, questi attivasse uno degli oggetti presenti nella sua lista di delega. Per condividere la conoscenza tra prototipi, abbiamo introdotto la regola:

Obj[\_Slot->Part] :- Obj[delega->Deleg], Deleg[\_Slot->Part].

Questo meccanismo permette di condividere conoscenze fra prototipi anche non appartenenti alla stessa "famiglia".

### 3 Rappresentazione e apparato deduttivo per il ragionamento spaziale qualitativo

Uno degli obiettivi principali del lavoro è stato quello di voler definire un formalismo in grado di rappresentare i concetti di base e nello stesso tempo, le peculiarità, i dettagli e le *sfumature* degli artefatti complessi. Un tale tipo di formalismo, riteniamo, risulta adeguato per la realizzazione di algoritmi orientati alla visualizzazione di artefatti partendo da descrizioni qualitative. Nella descrizione qualitativa di una scena, le informazioni possono essere parziali e contenere degli errori. Il completamento delle descrizioni, o la correzione degli errori, possono essere effettuati (in modalità automatica o semiautomatica), solo se si è in grado di inserire, in un sistema specializzato a eseguire inferenze su relazioni spaziali (ragionatore), completi prototipi ai quali un ragionatore può fare riferimento. L'approccio computazionale che suggeriamo, quindi, è quello di avere descrizioni dettagliate ed *espressive* di istanze di classe, per le definizioni dei prototipi di riferimento. In maniera tale da permettere, per il problema del completamento, un confronto fra l'istanza che si vuole disegnare e il prototipo esistente nella base di conoscenza. La scoperta dei dati e delle relazioni mancanti, può essere in tal modo effettuata, da parte di un sistema che esegue il confronto sopra citato, in base all'assiomatica di relazioni spaziali definita (per maggiori dettagli di un tale sistema si veda avanti).

Le relazioni spaziali qualitative di base da noi scelte, per la costruzione dell'assiomatica, sono le direzionali e le topologiche.

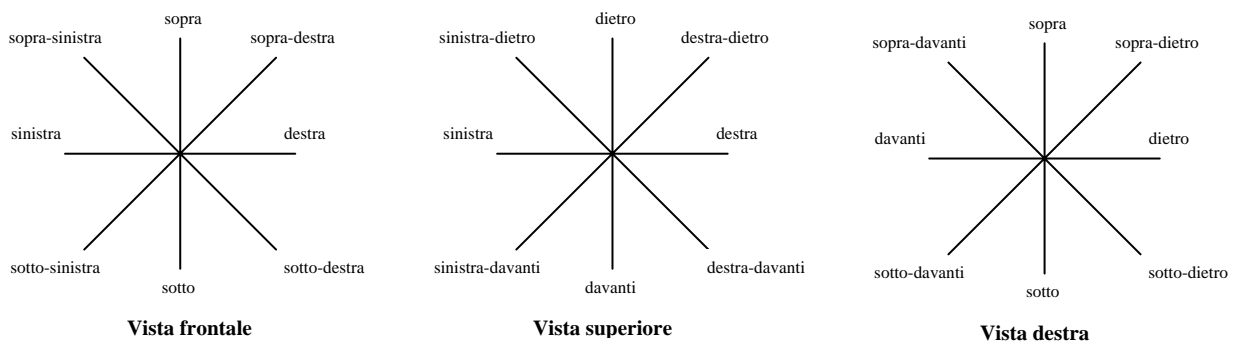
#### 3.1 Relazioni direzionali

La scelta dell'insieme di relazioni spaziali direzionali è ricaduta su alcune relazioni per le quali c'è poca ambiguità in termini interpretativi (semantici), ma soprattutto perché, tali relazioni, permettono di descrivere la scena dal punto di vista dell'osservatore. I concetti di base scelti sono:

RelDir = {sopra, sotto, destra, sinistra, davanti, dietro}

Tali relazioni di base possono essere utilizzate nelle istanze congiuntamente (eccetto conflitti) per rappresentare altre relazioni direzionali. Nello schema che segue mostriamo le combinazioni che si possono ottenere combinando le relazioni di base RelDir.

Alle relazioni qualitative RelDir sono state associate alcune proprietà: la transitività e la dualità.



trans(R1-R2):Rel[obj1->X, obj2->Z] :-  
 R1:Rel[obj1->X, obj2->Y], R2:Rel[obj1->Y, obj2->Z].

dual(R): Relinv[obj1->X, obj2->Y] :-  
 R:Rel[obj1 -> Y, obj2 -> X], inv(Rel,Relinv).

Il primo predicato inferisce che, esiste una relazione Rel tra X e Z, se esiste Rel tra X e Y, e Rel Y e Z (transitività). La seconda inferisce che esiste una relazione Relinv tra X e Y, se esiste una relazione Rel tra Y e X, e una relazione inv(Rel, Relinv) (Relinv è la relazione inversa della relazione Rel). Per ciascuna relazione spaziale qualitativa direzionale, sono state definite le inverse:

inv(sopra,sotto), inv(sotto,sopra), inv(sinistra,destra),  
 inv(destra,sinistra), inv(davanti,dietro), inv(dietro,davanti).

Le proprietà delle relazioni qualitative sono state definite nell'assiomatica come predicati del secondo ordine (adoperando quantificazioni sui predicati). Tale scelta è giustificata dal fatto che nell'apparato di deduzione è possibile estendere tutte le relazioni spaziali qualitative di base (non solo quelle direzionali), senza la necessità di aggiungere per ogni nuova relazione altre regole.

In aggiunta, alle relazioni direzionali precedenti, abbiamo definito altre relazioni che permettono di descrivere l'allineamento relativo tra oggetti, sfruttando la conoscenza della loro posizione e dimensione. Le relazioni sono:

RelAlign = {allineato\_inalto, allineato\_inbasso, allineato\_adestra, allineato\_asinistra,  
 allineato\_suldavanti, allineato\_suldietro}.

Anche per queste ultime relazioni qualitative, abbiamo associato le proprietà di simmetria e di transitività:

simm(R):Rel\_allineamento[obj1->X, obj2->Y] :-  
 R:Rel\_allineamento[obj1->Y, obj2->X].

trans(R1-R2): Rel\_allineamento[obj1 -> X, obj2 -> Z] :-  
 R1: rel\_allineamento[obj1 -> X, obj2 -> Y],  
 R2: rel\_allineamento[obj1 -> Y, obj2 -> Z], X\=Z.

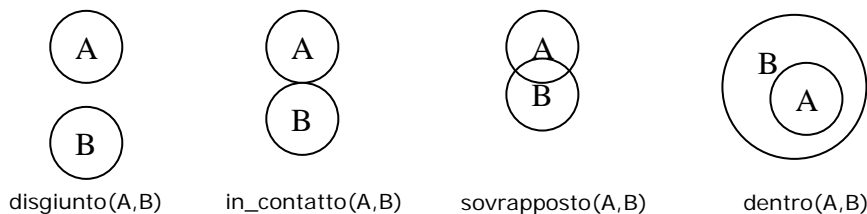
La prima relazione permette di inferire che esiste un allineamento tra gli oggetti X e Y, se esiste un allineamento tra Y e X. La seconda permette di inferire che esiste un allineamento, tra gli oggetti X e Z, se esistono gli allineamenti tra X e Y, e tra Y e Z (con X diverso da Z).

### 3.2 Relazioni topologiche

Per il formalismo proposto, l'insieme delle relazione topologiche di base è il seguente:

RelTopol = {disgiunto, in\_contatto, sovrapposto, dentro}

Nella figura seguente diamo una caratterizzazione di tali relazioni topologiche:



La proprietà di simmetria delle relazioni `disgiunto`, `in_contatto` e `sovrapposto`, sono state definite nel modo seguente:

```
simm(R):disgiunto[obj1->X, obj2->Y] :-
    R:disgiunto[obj1->Y, obj2->X].
simm(R):sovrapposto[obj1->X, obj2->Y] :-
    R:sovrapposto[obj1->Y,obj2->X].
simm(R): in_contatto [obj1->X, obj2->Y] :-
    R: in_contatto [obj1->Y, obj2->X].
```

La proprietà transitiva della relazione `dentro`:

```
trans(R1-R2):dentro[obj1->X, obj2->Z] :-
    R1:dentro[obj1->X, obj2->Y],
    R2:dentro[obj1->Y, obj2->Z].
```

Per le relazioni `dentro` e `sovrapposto` abbiamo definito una regola di composizione che permette di derivare che due oggetti X e Z sono sovrapposti, se Y è dentro un oggetto X, e Y è sovrapposto a Z.

```
comp(R1-R2):sovrapposto[obj1->X, obj2->Z] :-
    R1:dentro[obj1->Y, obj->X],
    R2:sovrapposto[obj1->Y, obj2->Z].
```

Per le relazioni `dentro` e `disgiunto`, invece, abbiamo definito una regola di composizione che permette di derivare che due oggetti X e Z sono disgiunti se X è dentro un oggetto Y e Y è disgiunto da Z.

```
comp(R1-R2):disgiunto[obj1->X, obj2->Z] :-
    R1:dentro[obj1->X, obj2->Y], R2:disgiunto[obj1->Y, obj2->Z].
```

### 3.3 Proprietà algebriche delle relazioni: idempotenza, antiriflessiva e composizione delle inverse

Nel formalismo proposto abbiamo implementato le proprietà algebriche di idempotenza, di antiriflessività e composizione delle inverse mediante le seguenti regole<sup>4</sup>:

```
idempotenza():-
    while((R1:Rel[obj1->X,obj2->Y], R2:Rel[obj1->X,obj2->Y], (R1\=R2)))
    do ( delete{R2:Rel} ).
```

```
antiriflessiva() :-
    deleteall{ R:Dir | R:Dir[obj1->X, obj2->X] }.
```

```
cancella_inverse() :-
    deleteall{R1:rel(Rel1,Dist), R2:rel(Rel2,Dist) |
        R1:rel(Rel1,Dist)[obj1->X, obj2->Y],
        R2:rel(Rel2,Dist)[obj1->X, obj2->Y],
```

<sup>4</sup> I predicati extra-logici presenti in tali regole sono costruiti dell'implementazione utilizzata Flora2: *while(Condition)...do(Action)* è semanticamente identico al costrutto dei linguaggi imperativi, cerca tutte le derivazioni nelle quali è verificata la *Condition* ed esegue l'*Action*. *Deleteall*, come *delete*, *insertall* ed *insert* sono predicati Flora2, mediante i quali è possibile, rispettivamente: cancellare e inserire tutti o uno i predicati passati come parametro (similmente al *retract* e *assert* del Prolog)

```

Rel1::rel_direz, Rel2::rel_direz,
inv(Rel1,Rel2), Dist::dist_qual),
deleteall{R1:rel(Rel,Dist), R2:rel(Rel,Dist) |
R1:rel(Rel,Dist)[obj1->X, obj2->Y],
R2:rel(Rel,Dist)[obj1->Y, obj2->X],
Rel::rel_direz, Dist::dist_qual}.

```

(l'idempotenza elimina tutte le relazioni uguali lasciandone una sola, l'antiriflessiva elimina tutte le relazioni che sono applicate al medesimo oggetto, la composizione delle inverse elimina tutte le relazioni che sono una l'inversa dell'altra).

### 3.4 Distanza qualitativa e sue proprietà

Nell'assiomatica di ragionamento qualitativo, abbiamo adottato una nozione di distanza qualitativa definita mediante sei valori qualitativi di riferimento:

$$D = \{\text{moltovicino, vicino, media, lontano, moltolontano}\}$$

Tali valori sono stati scelti come traduzione dei valori utilizzati nei sistemi *multi-step distance* presentati in [8] nel quale sono utilizzati i simboli *C, c, f, F* come abbreviazioni di *close(vicino)* e *far(lontano)*, ed utilizzando le lettere maiuscole e le minuscole per enfatizzare la più o meno vicinanza o lontananza. In aggiunta abbiamo inserito un valore mediano per una raffinazione ulteriore.

In corrispondenza di tali valori, il ragionatore da noi definito, per il ragionamento spaziale qualitativo, associa distanze (quantitative) in funzione del particolare contesto, questo tipo di intuizione è presente in [17]. Il contesto, nel nostro caso, viene stabilito dai tipi, dalle dimensioni e dalle relazioni (direzionali) degli oggetti presenti nella scena. La distanza media è calcolata dalle dimensioni dei BoundingBox dei due oggetti X (Xweight, Xheight, Xdepth) e Y (Yweight, Yheight, Ydepth) interessati dalla relazione, utilizzando la regola (le altre calcolate in funzione di tale regola):

```

dist[valDist->DistMedia, obj1->X, obj2->Y] :-
    boundingboxX[wbb->Xweight, hbb->Xheight, dbb->Xdepth ],
    boundingboxY[wbb->Yweight, hbb->Yheight, dbb->Ydepth ],
    DimBBX = ((Xweight+Xheight+Xdepth)/3.0),
    DimBBY = ((Yweight+Yheight+Ydepth)/3.0),
    DistMedia = ((DimBBX + DimBBY)/2.0).

```

Per la nozione di distanza qualitativa, abbiamo definito un ordinamento tra distanze (qualitative) e un'operazione di addizione di distanze addDistq/3.

Abbiamo utilizzato il predicato succ(D1,D2) per indicare che "una distanza D1 segue nell'ordinamento una distanza D2", ordinando i valori qualitativi di riferimento nel seguente modo:

$$\text{succ}(\text{moltolontano, lontano}), \text{succ}(\text{lontano, media}), \text{succ}(\text{media, vicino}), \text{succ}(\text{vicino, moltovicino})$$

A partire da tali relazioni di precedenza, abbiamo definito il predicato max/3 che calcola la distanza più grande tra due distanze qualitative:

```

max(X,X,X):- !.
max(X,Y,X) :- succ(X,Y), !.
max(X,Y,Y) :- succ(Y,X), !.
max(X,Y,X) :- succ(X,Z), max(Z,Y,Z),!.
max(X,Y,Y) :- succ(Y,Z), max(Z,X,Z),!.

```

In altri approcci relativi la gestione delle distanze qualitative[8] sono suggeriti tre diversi metodi di gestire la composizione dei valori di distanza, siamo dell'opinione che non esista una scelta migliore universalmente, ma piuttosto riteniamo che a seconda del dominio sia necessario scegliere l'approccio che meglio lo approssima. La definizione di somma di due distanze qualitative, da noi utilizzata,  $\text{addDistq}(D1, D2, \text{Somma})$  con  $D1$  e  $D2 \in D$ , è:

$\text{addDistq}(D1, D2, R) :- R = \text{moltolontano}, !.$   
 $\text{addDistq}(D1, D2, R) :- \max(D1, D2, R1), (R1 \neq \text{moltolontano}, \text{succ}(R, R1)), !.$

Alcuni esempi di applicazione di tale definizione sono:

?-  $\text{addDistq}(\text{moltovicino}, \text{moltovicino}, R) \rightarrow R = \text{vicino}$   
 ?-  $\text{addDistq}(\text{moltovicino}, \text{lontano}, R) \rightarrow R = \text{moltolontano}$

### 3.5 Inferenze fra direzioni e distanze

Per costruire connessioni fra direzioni e distanze del tipo  $\text{rel}(\text{sopra}, \text{vicino})$  e  $\text{rel}(\text{destra}, \text{lontano})$ , abbiamo introdotto il predicato del secondo ordine  $\text{rel}(\text{Rel}, \text{Dist})[\text{obj1} \rightarrow X, \text{obj2} \rightarrow Y]$  a partire dal quale è possibile definire regole estremamente compatte, come la regola  $\text{dual}$ :

$\text{dual}(R) : \text{rel}(\text{Rel}, \text{Dist})[\text{obj1} \rightarrow X, \text{obj2} \rightarrow Y] :-$   
 $R : \text{rel}(\text{Relinv}, \text{Dist})[\text{obj1} \rightarrow Y, \text{obj2} \rightarrow X], \text{inv}(\text{Rel}, \text{Relinv}).$

Una possibile istanziiazione della regola appena riportata è:

$\text{id} : \text{rel}(\text{sopra}, \text{vicino})[\text{obj1} \rightarrow \text{capitello}, \text{obj2} \rightarrow \text{fusto}] \rightarrow$   
 $\text{dual}(\text{id}) : \text{rel}(\text{sotto}, \text{vicino})[\text{obj1} \rightarrow \text{fusto}, \text{obj2} \rightarrow \text{capitello}]$

Come si può notare, l'adozione di regole del secondo ordine, permette un notevole risparmio di definizioni di regole. Quantificando sulle variabili della regola, infatti, è possibile ragionare su tutti i predicati di relazioni spaziali e su tutti quelli delle distanze qualitative.

Le relazioni sono transitive lungo la stessa direzione e stessa distanza, applicando la regola di somma precedentemente introdotta:

$\text{trans}(R1-R2) : \text{rel}(\text{Rel}, \text{DistNew})[\text{obj1} \rightarrow X, \text{obj2} \rightarrow Z] :-$   
 $R1 : \text{rel}(\text{Rel}, \text{Dist})[\text{obj1} \rightarrow X, \text{obj2} \rightarrow Y],$   
 $R2 : \text{rel}(\text{Rel}, \text{Dist})[\text{obj1} \rightarrow Y, \text{obj2} \rightarrow Z],$   
 $(X \neq Z), \text{succ}(\text{DistNew}, \text{Dist}).$

La seconda regola, invece, si applica a relazioni con la stessa direzione ma a distanze diverse:

$\text{trans}(R1-R2) : \text{rel}(\text{Rel}, \text{Dist2})[\text{obj1} \rightarrow X, \text{obj2} \rightarrow Z] :-$   
 $R1 : \text{rel}(\text{Rel}, \text{Dist1})[\text{obj1} \rightarrow X, \text{obj2} \rightarrow Y],$   
 $R2 : \text{rel}(\text{Rel}, \text{Dist2})[\text{obj1} \rightarrow Y, \text{obj2} \rightarrow Z],$   
 $(X \neq Z), \text{Rel} :: \text{rel\_direz}, \text{Dist1} :: \text{dist\_qual},$   
 $\text{Dist1} :: \text{dist\_qual}, \max(\text{Dist2}, \text{Dist1}).$

Se esistono due relazioni uguali applicate una a X e Y con distanza Dist1, l'altra a Y e Z con Dist2, con  $\text{Dist2} \Rightarrow \text{Dist1}$ , si può inferire che X è in relazione con Z con distanza Dist2.

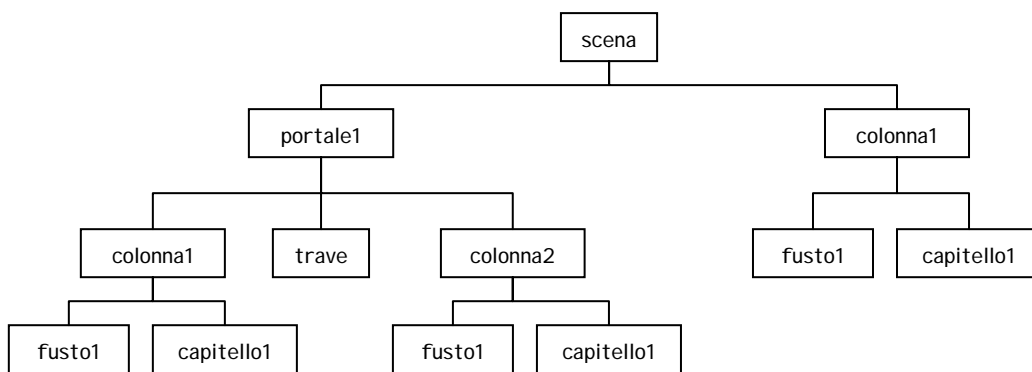
## 4 Applicazione al problema della visualizzazione di relazioni spaziali qualitative

L'applicazione che proponiamo, realizza il completamento di oggetti, delle relative relazioni spaziali qualitative, delle distanze tra gli oggetti, e la generazione delle coordinate necessarie per il disegno della scena: realizzando una visione tridimensionale della scena descritta mediante il formalismo introdotto.

Il primo passo è quello di effettuare il completamento dell'istanza confrontandola con l'ontologia di riferimento. Il formalismo definito permette di introdurre classi che vincolano fortemente le istanze. Ciò permette di completare le istanze degli oggetti che si cerca di visualizzare. Il completamento interessa sia le parti dell'oggetto, sia le relazioni spaziali qualitative tra di esse. Le informazioni dell'ontologia coprono anche informazioni riguardanti le dimensioni degli oggetti primitivi, e la primitiva da utilizzare per il rendering. Combinando queste informazioni, con le caratteristiche derivate durante il calcolo degli oggetti composti, è possibile definire geometricamente la scena.

#### 4.1 Procedura di visualizzazione

L'istanza, che descrive la scena, è rappresentata mediante un albero. La scena è analizzata, in maniera ricorsiva, per singolo nodo.



Si analizza ogni nodo nel suo sistema di riferimento (cartesiano) locale, partendo dai nodi che hanno come figli, solo foglie (oggetti primitivi). Successivamente, sono calcolate le coordinate di ogni oggetto (figlio del nodo considerato), applicando le proprietà e gli assiomi precedentemente introdotti e applicando regole di conversione per passare dalle relazioni spaziali qualitative, alle coordinate di posizionamento degli oggetti. Riportiamo un sottoinsieme di tali regole:

- I st:rel(destra,Dist)[obj1->A,obj2->B] :-  $X(A) = X(B) + \text{Dist}$ .
- I st:rel(sinistra,Dist)[obj1->A,obj2->B] :-  $X(A) = X(B) - \text{Dist}$ .
- I st:rel(sopra,Dist)[obj1->A,obj2->B] :-  $Y(A) = Y(B) + \text{Dist}$ .
- I st:rel(sotto,Dist)[obj1->A,obj2->B] :-  $Y(A) = Y(A) - \text{Dist}$ .
- I st:rel(dietro,Dist)[obj1->A,obj2->B] :-  $Z(A) = Z(B) + \text{Dist}$ .
- I st:rel(davanti,Dist)[obj1->A,obj2->B] :-  $Z(A) = Z(B) - \text{Dist}$ .

Le coordinate di un oggetto A sono calcolate da quelle dell'oggetto B sommando (sottraendo) la distanza Dist che li separa. Seguendo questa metodologia, partendo da un oggetto posto al centro della scena, si posizionano tutti gli altri oggetti componenti il nodo considerato. Terminato il posizionamento, si calcola il BoundingBox del nodo e si itera il calcolo sino alla radice dell'albero. Posizionati, infine, localmente i singoli oggetti si effettua un calcolo che permette il posizionamento globale nella scena.

In questo lavoro abbiamo realizzato un sistema per la visualizzazione delle relazioni spaziali qualitative, secondo l'approccio sopra riportato. Nello schema seguente riportiamo in maniera schematica i passi del funzionamento di tale sistema.

```

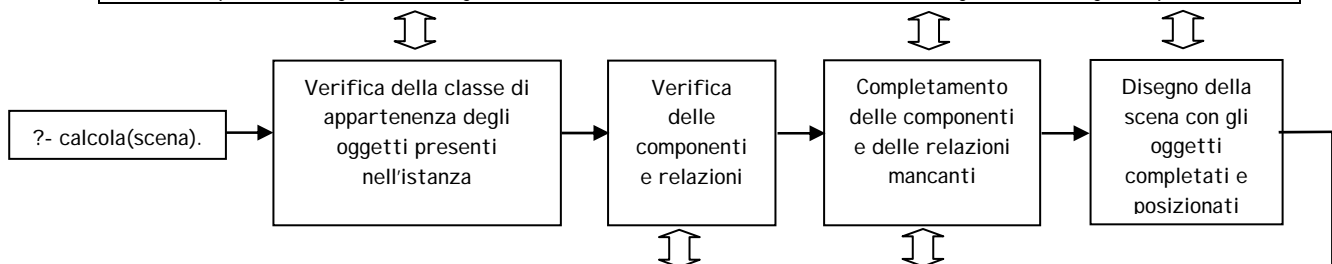
Istanza di scena

scena[formato_da->{portale1,portale2},
      relazioni_spaziali->{dxpp1,dipp1}].
dxpp1:rel(destra,vicino)[obj1->portale1, obj2->portale2].
dipp1:rel(dietro,vicino)[obj1->portale1, obj2->portale2].
portale1:portali_u_cap.
portale2: portali_u_cap.
portale1[formato_da->{trave,col1,col2},
          relazioni_spaziali ->{sot1,sot2,all1}].
portale2[formato_da->{trave,col1,col2},
          relazioni_spaziali ->{sot1,sot2,all1}].
trave:tx.
col1:c1.
col2:c2.
sot1:rel(sopra,zero)[obj1->trave, obj2->col1].
sot2:rel(sopra,zero)[obj1->trave, obj2->col2].
all1:allineato_suldietro[obj1->col2, obj2->col1].

col1[formato_da->{fusto1, capitello1},
      relazioni_spaziali->{}].
fusto1:fusto.
capitello1:capitello.
sotto1:rel(sotto,zero)[obj1->fusto1, obj2->capitello1].

col2[formato_da->{fusto1, capitello1},
      relazioni_spaziali ->{sotto1}].
fusto1:fusto.
capitello1:capitello.
sotto1:rel(sotto,zero)[obj1->fusto1, obj2->capitello1].

```



```

Ontologia di riferimento

/* Definizione classe Portale Pi-greco */
portali_pigreco::portali.
portale_pigreco1:portali_pigreco.
portale_pigreco1[relazioni_spaziali -> {s1, s2, i1, i2},
                 delega -> {portale1}].

/* Definizione classe Portale ad u capovolta */
portali_u_cap::portali_pigreco.
portale_u_cap1: portali_u_cap.
portale_u_cap1[relazioni_spaziali -> {adx1, asx1},
               delega -> {portale_pigreco1}].

/* Relazioni spaziali la classe portali Pi-greco*/
s1:sopra[obj1 -> tx, obj2 -> c1].
s2:sopra[obj1 -> tx, obj2 -> c2].
i1:in_contatto[obj1 -> tx, obj2-> c1].
i2:in_contatto[obj1 -> tx, obj2-> c2].

/* Relazioni spaziali per la classe portali_u_cap*/
asx1:allineato_asinistra[obj1 -> tx, obj2 -> c1].
adx1:allineato_adestra[obj1 -> tx, obj2 -> c2].

```



← Rendering 3D

## 4.2 Nota sull'architettura software

Il sistema di visualizzazione di relazioni spaziali qualitative realizzato ha reso necessario l'integrazione di due ambienti di programmazione: Flora2, implementazione di Frame Logic basata su XSB [5] (un sistema Prolog OpenSource) e Java, messi in connessione mediante un bridge denominato Java2Flora, che permette effettuare *interrogazioni* Flora2 da Java. Per la realizzazione e il disegno della scena sono state utilizzate le API Java3D della SUN. Java3D è una libreria in grado di effettuare disegno tridimensionale implementando le caratteristiche dei moderni motori tridimensionali uniti alle *capacità* del linguaggio Java.

## 5 Lavori di riferimento e discussione

Nel lavoro [3] è stato proposto un formalismo per la rappresentazione di strutture architettoniche classiche. In tale lavoro si è affrontato il problema di rappresentare relazione meronomiche parte-totalità utilizzando un approccio esplicito. Il formalismo permette di esprimere rappresentazioni del tipo:

<Concept identifier (187)> <Synonyms (colonna)> <Superconcept (159)> <Form (132)> <Stuff (158)> <Components (361 (exactly 1)) (362 (exactly 1)) (37 (exactly 1))> <Position (upon 362 361) (upon 37 362)>  
<Glossa (Piedritto a sezione circolare composto di base, fusto monolitico o a segmenti cilindrici, capitello, con funzione portante o decorativa)> con 159 “piedritto”, 361 “base”, 362 “fusto”, 37 “capitello”, 53 “trabeazione”, 132 “cilindro”, 158 “pietra”.

Tale formalismo è stato definito nello specifico settore linguistico per l'archeologia, e riteniamo che poca attenzione sia stata data agli aspetti deduttivi-computazionali. La rappresentazione appare avere, comunque, qualche limite in termini di *espressività*. Con tale formalismo, ad esempio, non risulta possibile definire relazioni spaziali differenti per parti che appartengono ad una stessa categoria.

Relativamente ad altri formalismi per il ragionamento spaziale come [18][19][21] riteniamo che le informazioni, sulle quali si effettuano ragionamenti, siano insufficienti nel dominio di interesse del nostro lavoro, tali formalismi (denominati RCC) si occupano prevalentemente di relazioni topologiche che risultano essere inutilizzabili per ricreare la configurazione di una scena in tre dimensioni. Infatti, una relazione topologica ci può *indicare* dove un oggetto non si trova, ma non dove esso si trova. Dire che A è disgiunto da B, ci informa del fatto che A non è a contatto con B, ma che si può trovare in qualsiasi punto della scena. Questo è poco utile nel dominio da noi analizzato.

Riguardo al problema del ragionamento spaziale qualitativo, alcuni lavori [9] si sono occupati degli oggetti estesi (non puntiformi), utilizzando relazioni topologiche (*adiacenti, sovrapposti,...*), non affrontando, però, il problema delle distanze. Tale approccio è stato seguito anche da altri [8]. Altri lavori (relativi a sistemi GIS) hanno cercato di *catturare* il ragionamento spaziale qualitativo mediante rappresentazioni algebriche [7][10]. In [7] è proposto un approccio dove si definiscono le relazioni spaziali, e le loro inferenze, in un scenario spaziale composto di oggetti di tipo contenitori e superfici. Il lavoro suggerisce una metodologia di approccio generale al problema del ragionamento spaziale. Purtroppo esso si occupa soltanto di due tipologie di oggetti (contenitori e superfici). L'investigazione delle deduzioni possibili è completa, ma presenta il problema di non considerare né le dimensioni, né la categoria di appartenenza di ciascun oggetto.

## 6 Ringraziamenti

Ringraziamo Giovanni Criscuolo e Nicola Guarino per gli utilissimi suggerimenti che ci hanno fornito in questo lavoro. Uno speciale ringraziamento va a Antonio Sorgente per il suo aiuto su alcune definizioni dell'apparato inferenziale realizzato.

## 7 Bibliografia

- [1] A. Artale, E. Franconi, N. Guarino, L. Pazzi, *Part-Whole Relations in Object-Centered Systems: An overview*, Data & Knowledge Engineering (DKE) journal 20 347-383 – North-Holland, Elsevier, 1996.
- [2] N. Guarino, *Formal Ontology in Information Systems*, Proceedings of FOIS'98, Trento, Italy. Amsterdam, IOS Press, 6-8 June 1998.
- [3] Cappelli A., Novella Catarsi A., Nichelassi P., Moretti L., *Un formalismo per rappresentare strutture architettoniche classiche*, Atti del Convegno, Contesti Virtuali e Fruizione dei Beni Culturali, Napoli, 22-23 Maggio 2003.
- [4] A. Calabrese, F. Mele, L. Serino, A. Sorgente, O. Talamo, *Rappresentazioni di conoscenze spaziali di siti archeologici*, AI\*IA 2003 - Ottavo Congresso Nazionale dell'AI\*IA, Polo didattico "L. Fibonacci", Università di Pisa, 23-26 Settembre 2003.
- [5] XSB Project v2.6 - <http://xsb.sourceforge.net/>
- [6] Flora Project v0.92 - <http://flora.sourceforge.net/>

- [7] M. Egenhofer and A. Rodríguez, *Relation Algebras over Containers and Surfaces: An Ontological Study of a Room Space*, Journal of Spatial Cognition and Computation, Vol. 1, No. 2, pp. 155-180, 1999.
- [8] A. U. Frank, *Qualitative Spatial Reasoning about Distances and Directions in Geographic Space*, Journal of Visual Languages and Computing 3, 343-371, 1992.
- [9] D. Papadias and T. Sellis, *Qualitative Representation of Spatial Knowledge in Two-Dimensional Space* VLDB Journal, 3, 479-516, Ralf Hartmut Gfiting Editor, 1994.
- [10] C. Eschenbach and L. Kulik, *An axiomatic approach to the spatial relations underlying left-right and in front of-behind*, KI-97 Advances in Artificial Intelligence (pp. 207 – 218). Berlin: Springer, 1997.
- [11] D. M. Mark, S. Svorou and D. Zubin, *Spatial terms and spatial concepts; geographic, cognitive, and linguistic perspectives*, International Geographic Information Systems (IGIS) Symposium, 1987.
- [12] G. Retz-Schmidt, *Various views on spatial prepositions*, AI Magazine, 9, 95–105, 1988
- [13] W.J.M. Levelt, *Some perceptual limitations on talking about space*, A.J. van Doorn, W.A. van der Grind & J.J. Koenderink (eds.): Limits in Perception, 1984.
- [14] R. Casati, C. V. Achille, *I trabocchetti della rappresentazione spaziale*, Sistemi Intelligenti 11:1, 1999.
- [15] M. Kifer, G. Lausen, J. Wu, *Logical Foundations of Object-Oriented and Frame-Based Languages*, Journal of ACM, 1995.
- [16] H. Lieberman, *Using prototypical objects to implement shared behavior in object oriented systems*, Artificial Intelligence Laboratory Massachusetts Institute of Technology Cambridge, 1986.
- [17] M. Worboys, *Nearness relations in environmental space*, Journal of GIS, 2001.
- [18] B. Bennett, *Logical Representations for Automated Reasoning about Spatial Relationships* PhD thesis, School of Computer Studies, University of Leeds, 1997.
- [19] A.G. Cohn and S.M. Hazarika, *Qualitative spatial representation and reasoning: an overview*, Fundamenta Informaticae, 45:1--29, 2001.
- [20] J. Renz and B. Nebel, *On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus*, Artificial Intelligence, 108(1-2), 1999
- [21] M. Wessel, *On Spatial Reasoning with Description Logics*, Proceedings of DL, 2002.
- [22] A. Sorgente, *Rappresentazione di relazioni spaziali mediante ontologie*, Tesi di laurea in Informatica, Università di Napoli Federico II, a.a. 2002-2003

## 8 Appendice A : BNF del formalismo

Presentiamo la BNF del formalismo proposto. Il formalismo si compone della definizione degli oggetti nella base di conoscenza e la descrizione della scena e delle istanze.

### BNF

```

Start ← Def_sottoclasse | Def_istanza | Def_istanza_ogg_semplice | Def_istanza_ogg_composto | Def_istanza_rel_spaziali.
/***** Definizione classe-sottoclasse *****/
Def_sottoclasse ← Def_obj_semplici | Def_obj_composti | Def_rel_spaziali.
Def_obj_semplici ← Id :: IdOS | Id :: obj_semplici.
Def_obj_composti ← Id :: IdOC | Id :: obj_composti.
Def_rel_spaziali ← Id :: IdRel | Id :: Relazioni_spaziali.
/***** Definizione istanza *****/
Def_istanza ← Def_ist_obj_semplici | Def_ist_obj_composti | Def_ist_rel_spaziali.
Def_ist_obj_semplici ← IdIOS : IdOS.
Def_ist_obj_composti ← IdIOC : IdOC.
Def_ist_rel_spaziali ← IdIRel : IdRel | IdIRel : Relazioni_spaziali.
/***** Definizione classe oggetto semplice *****/
Def_classe_ogg_semplice ← IdOS[Def_slot_proprieta_ogg_semplice].
Def_slot_proprieta_ogg ← Forma_ogg, Dimensioni_BB, Dimensioni_ogg, Rotazioni_ogg |
                        Forma_ogg, Dimensioni_BB, Rotazioni_ogg |
                        Forma_ogg, Dimensioni_BB, Dimensioni_ogg |
                        Forma_ogg, Dimensioni_BB.
Forma_ogg ← ha_forma, *=> Forme.
Dimensioni_BB ← hBB *=> float, wBB*=> float, dBB*=> float.
Dimensioni_ogg ← dimensione *=> Dimensioni.

```

```

Rotazioni_ogg ← rotazione *=> Rotazioni.
/***** Definizione istanza oggetto semplice *****/
Def_classe_ogg_semplice ← IdOS[Def_slot_proprieta_ist_ogg_semplice].
Def_slot_ist_proprieta_ogg ← Forma_ist_ogg, Dimensioni_ ist_BB, Dimensioni_ ist_ogg, Rotazioni_ ist_ogg|
                             Forma_ ist_ogg, Dimensioni_ ist_BB, Rotazioni_ ist_ogg |
                             Forma_ ist_ogg, Dimensioni_ ist_BB, Dimensioni_ ist_ogg|
                             Forma_ ist_ogg, Dimensioni_ ist_BB.
Forma_ist_ogg ← ha_forma, -> Forme.
Dimensioni_ ist_BB ← hBB -> float, wBB -> float, dBB -> float.
Dimensioni_ ist_ogg ← dimensione -> Dimensioni.
Rotazioni_ ist_ogg ← rotazione -> Rotazioni.
/***** Definizione istanza oggetto composto *****/
Def_istanza_ogg_composto ← IdOC[ Def_slot_proprieta_ist_ogg_composto ].
Def_slot_proprieta_ist_ogg_composto ←
    Def_componenti, Def_rel_spaziali, Dimensioni_ogg, Rotazioni_ogg, Delega_ogg |
    Def_componenti, Def_rel_spaziali, Rotazioni_ogg, Delega_ogg |
    Def_componenti, Def_rel_spaziali, Dimensioni_ogg, Delega_ogg |
    Def_componenti, Dimensioni_ogg, Rotazioni_ogg, Delega_ogg |
    Def_componenti, Rotazioni_ogg, Delega_ogg |
    Def_componenti, Dimensioni_ogg, Delega_ogg |
    Def_componenti, Delega_ogg.
    Def_componenti.
Def_componenti ← formato_da -> { I stanze_oggetti }.
Def_rel_spaziali ← relazioni_spaziali -> { I stanze_rel_spaziali }.
I stanze_oggetti ← I stanza_oggetto | I stanza_oggetto, I stanze_oggetti.
I stanze_rel_spaziali ← I stanza_rel_spaziale | I stanza_rel_spaziale, I stanze_rel_spaziali.
Dimensioni_ogg ← dimensione -> Dimensioni.
Rotazioni_ogg ← rotazione -> Rotazioni.
Delega_ogg ← delega -> { I stanze_oggetti }.

I stanza_oggetto ← IdIOS | IdIOC.
I stanza_rel_spaziale ← IdRel.
/***** Definizione classe relazione spaziale qualitativa *****/
Def_classe_rel_spaziali ← IdRel [ Def_obj1_rel_spaz, Def_obj2_rel_spaz ].
Def_obj1_rel_spaz ← obj1 *=> Classe_oggetto.
Def_obj2_rel_spaz ← obj2 *=> Classe_oggetto.
/***** Definizione generale simboli non-terminali e terminali *****/
Forme ← rectangle | box | cylinder | sphere | ...
Dimensioni ← moltopiccolo | piccolo | normale | grande | moltogrande.
Rotazioni ← ruotato_asinistra | ruotato_adestra | ... | inclinato_inavanti | inclinato_asinistra | ... | sottosopra | notdef.
Relazioni_spaziali ← rel(destra, moltovicino) | rel(destra,vicino) | ... | rel(sotto,vicino) | ... | disgiunto | in_contatto | ... |
                    allineato_adestra | ...
Id ← stringa con primo carattere minuscolo.
IdOS ← stringa. (nome di un oggetto semplice).
IdOC ← stringa. (nome di un oggetto composto).
IdRel ← stringa. (nome di una relazione spaziale).

```