Rappresentazione della conoscenza

Rappresentazione della conoscenza

Esigenze: rappresentare

- fatti complessi
- oggetti complessi, globalmente e/o nel dettaglio (meccanismi di focalizzazione)
- scenari
- sequenze di eventi

Caratteristiche di un buon sistema di rappresentazione:

- Adeguatezza rappresentativa: poter rappresentare tutti i tipi di conoscenza relativi ad un dominio
- Adeguatezza inferenziale: poter manipolare le strutture rappresentative in modo da inferire nuova conoscenza.
- Efficienza inferenziale: poter incorporare informazioni in più da usare come guida verso gli obiettivi nei meccanismi di inferenza (focalizzazione dell'attenzione/euristica)
- Efficienza nell'acquisizione: capacità di acquisire facilmente nuova conoscenza.

Classificazione delle tecniche:

metodi dichiarativi: la conoscenza viene rappresentata come collezione statica di fatti, affiancata da un piccolo insieme di procedure generali per la manipolazione (esempio: logica dei predicati).

metodi procedurali: la conoscenza viene rappresentata come procedure per il suo uso.

Vantaggi della rappresentazione dichiarativa:

- Ogni fatto va immagazzinato solo una volta, indipendentemente dal numero di modi diversi in cui può essere usato.

- È facile aggiungere nuovi fatti al sistema, senza cambiare né gli altri fatti né le procedure.

Vantaggi della rappresentazione procedurale:

- È facile rappresentare la conoscenza su come fare le cose
- È facile rappresentare la conoscenza che non si inserisce facilmente in molti schemi dichiarativi semplici, come ad esempio il ragionamento per default e quello probabilistico
- È facile rappresentare la conoscenza euristica su come fare le cose in modo efficiente

La maggior parte dei sistemi funzionanti usa una combinazione dei due metodi.

Esempio di applicazione: comprensione del linguaggio naturale.

Dalla comprensione di un testo come:

«*John decise di far visita a Bill. Guidò fino a casa sua, ma vide che tutte le luci erano spente e allora andò al cinema*».

Il sistema deve saper rispondere alle domande:

John vide Bill?

Di chi era la casa buia?

Chi andò al cinema?

Occorre che il sistema:

- possieda e sappia manipolare una grande quantità di conoscenza del mondo (*semantica* e *pragmatica*)
- conosca la *sintassi* e possieda il *vocabolario* del linguaggio

Osservazione: è desiderabile ma complicato tenere separati i due livelli (ma alcune interpretazioni sintattiche non sono possibili se non si conosce il contesto)

La descrizione delle strutture di conoscenza avviene mediante schemi.

Definizione: "Il termine *schema* si riferisce ad un'organizzazione attiva di relazioni passate, o di esperienze passate, che si deve sempre supporre siano operanti in ogni risposta organica adattiva,, (Bartlett, 1932).

Tipi di schemi:

- *Frame* (quadri), spesso usati per descrivere una collezione di attributi che un determinato oggetto, per esempio, una sedie, in genere possiede.
- *Script* (copioni), usati per descrivere sequenze di eventi comuni, come ad esempio ciò che succede quando si va al ristorante.
- *Stereotipi*, usati per descrivere insiemi di caratteristiche che spesso nelle persone sono presenti contemporaneamente.
- *Modelli a regole*, usati per descrivere caratteristiche comuni condivise all'interno di un insieme di regole in un sistema di produzioni.

Problemi connessi all'uso degli schemi:

- Esistono proprietà di oggetti così di base da essere presenti in quasi tutti i domini del problema? Se esistono, dovremo assicurarci che siano trattate adeguatamente in ognuno dei meccanismi che proponiamo. Se tali oggetti esistono, che cosa sono?

- A quale livello dovrebbe essere rappresentata la conoscenza? Esiste un buon insieme di primitive a cui possa essere riportata tutta la conoscenza? È utile usare primitive di questo tipo?

- Dato un grande quantitativo di conoscenza immagazzinata in una base di dati, come si può accedere alle parti rilevanti per i propri scopi quando è necessario?

Strutture di conoscenza basilari

Oggetti complessi → composizione di oggetti semplici

Classi complesse → composizione di classi più piccole

Come descrivere proprietà di oggetti comuni e quanto possibile universali?

Le primitive relazionali sono:

- Relazione ISA (cioè “is a”, “è un”): esprime una tassonomia gerarchica (quindi una categorizzazione):

CANE ISA ANIMALE-DOMESTICO

ANIMALE-DOMESTICO ISA ANIMALE

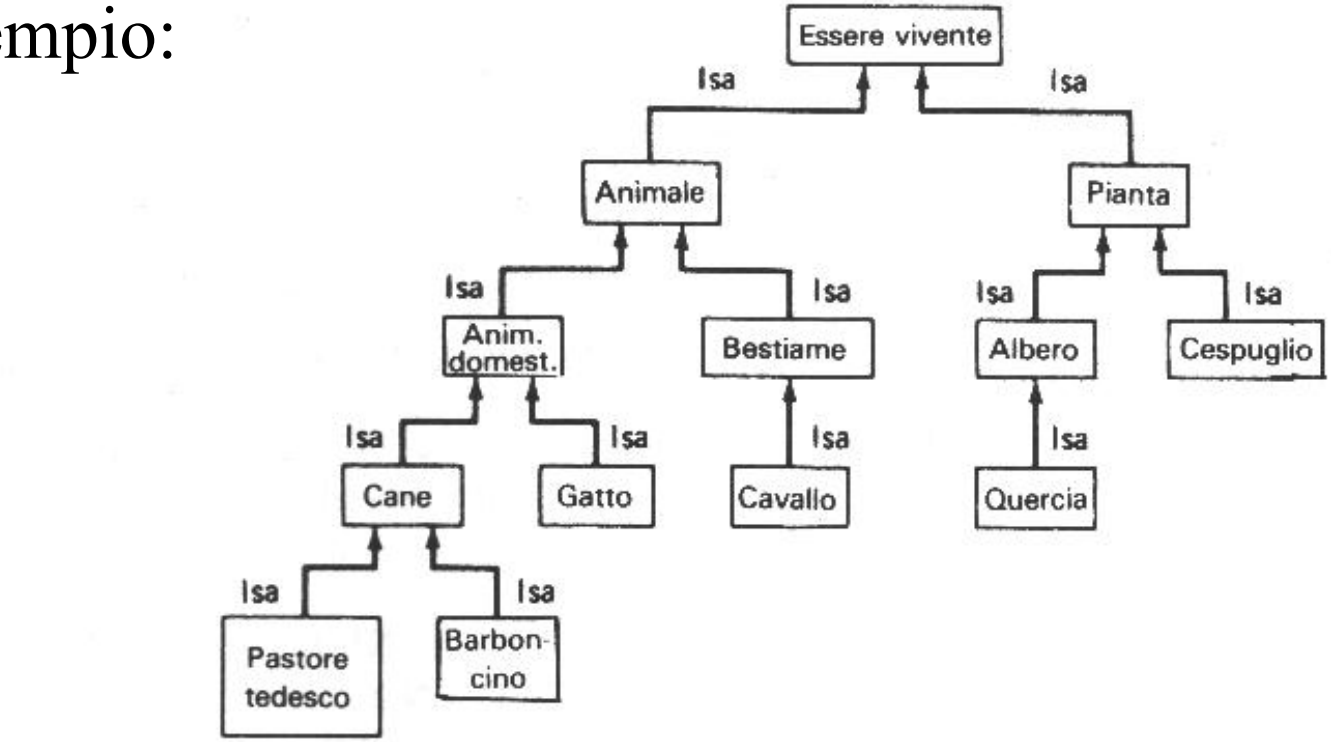
ANIMALE ISA ESSERE-VIVENTE

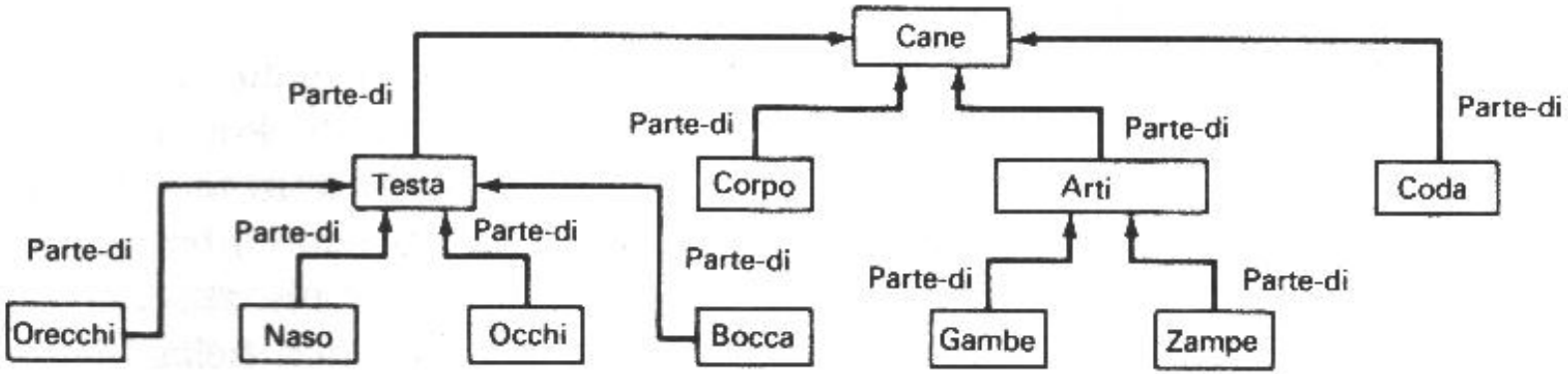
- Relazione PARTE-DI (PART-OF): esprime la composizione strutturale:

MANO PARTE-DI CORPO
DITO PARTE-DI MANO
UNGHIA PARTE-DI DITO

Entrambi sono ordinamenti parziali all'interno di un dominio. Al vertice di questo ordinamento c'è di solito un concetto molto generale (**entità**)

Esempio:


Essere vivente
Isa
Isa
Animale
Pianta
Isa
Isa
Isa
Isa
Anim. domest.
Bestiame
Albero
Cespuglio
Isa
Isa
Isa
Isa
Cane
Gatto
Cavallo
Quercia
Isa
Isa
Pastore tedesco
Barbon-cino


Cane
Parte-di
Parte-di
Parte-di
Parte-di
Testa
Corpo
Arti
Coda
Parte-di
Parte-di
Parte-di
Parte-di
Parte-di
Parte-di
Orecchi
Naso
Occhi
Bocca
Gambe
Zampe

Proprietà di queste relazioni

- transitività
- ereditarietà delle proprietà.

Poiché i cani hanno la coda, si può inferire che anche i barboncini hanno la coda. In che modo?

- Poiché bisogna tener conto delle eccezioni, si deve partire dal livello che rappresenta il concetto più specifico e risalire l'albero della struttura fino a trovare la proprietà cercata.

- Meglio ancora attuare un meccanismo di ragionamento per default (è più generale!)

Transitività:

Se vale:

BARBONCINO ISA CANE

CANE ISA ANIMALE-DOMESTICO

allora deve valere anche

BARBONCINO ISA ANIMALE-DOMESTICO

Ereditarietà delle proprietà

Le proprietà associate all'oggetto più generale possono essere attribuite all'oggetto più specifico (a meno che non si specifichi un'eccezione).

Attuazione:

Dipende dalle modalità scelte per rappresentare la conoscenza.

Esempio: Se si è scelta la logica dei predicati, le relazioni ISA e PART-OF sono predicati:

ISA(barboncino, cane)

ISA(cane, animale-domestico)

ISA(cavallo, bestiame)

Poi possiamo aggiungere al sistema l'enunciato:

$$\forall x \forall y \forall z \; ISA(x, y) \wedge ISA(y, z) \rightarrow ISA(x, z)$$

Ora possiamo provare facilmente che

ISA(Marco, romano)
o
ISA(barboncino, animale-domestico)

con tecniche note.

Esistono linguaggi specifici per queste applicazioni (esempio : KRL di Bobrow).

L'ereditarietà verrà approfondita in seguito.

Livello di rappresentazione

Le primitive IS-A e PART-OF non sono sufficienti a rappresentare il mondo completamente: ad esempio, non permettono di descrivere le azioni.

Quali e quante altre primitive introdurre? A quale livello di dettaglio?

In molte proposte l'approccio è di fornire rappresentazioni che tengano conto della semantica piuttosto che della sintassi.

Livello di rappresentazione

Le primitive IS-A e PART-OF non sono sufficienti a rappresentare il mondo completamente: ad esempio, non permettono di descrivere le azioni.

Quali e quante altre primitive introdurre? A quale livello di dettaglio?

In molte proposte l'approccio è di fornire rappresentazioni che tengano conto della semantica piuttosto che della sintassi.

Esempio: Fillmore propone la grammatica dei casi, basata sui concetti di agente ed oggetto.

La frase:

John ha dato un’occhiata a Sue

potrebbe essere rappresentata come :

ha_dato_un’occhiata(agente(John),oggetto(Sue))

Se poi si vuole rispondere alla domanda:

John ha visto Sue?

è sufficiente aggiungere un fatto come:

ha_dato_un'occhiata(x,y) → ha_visto(x,y)

In alternativa, si può correlare il concetto di

ha_dato_un'occhiata

al concetto di

vedere

così:

ha_visto(agente(John), oggetto(Sue),
lasso_di_tempo(breve))

ma si perdono le sfumature semantiche contenute in “dare un’occhiata” (questo fatto potrebbe essere accettabile).

Sembra che la conclusione sia del tipo:

“*è meglio trasformare tutti gli enunciati in una rappresentazione costruita su un piccolo insieme di primitive*”, quasi in forme canoniche.

Argomenti contro l'uso di primitive a basso livello:

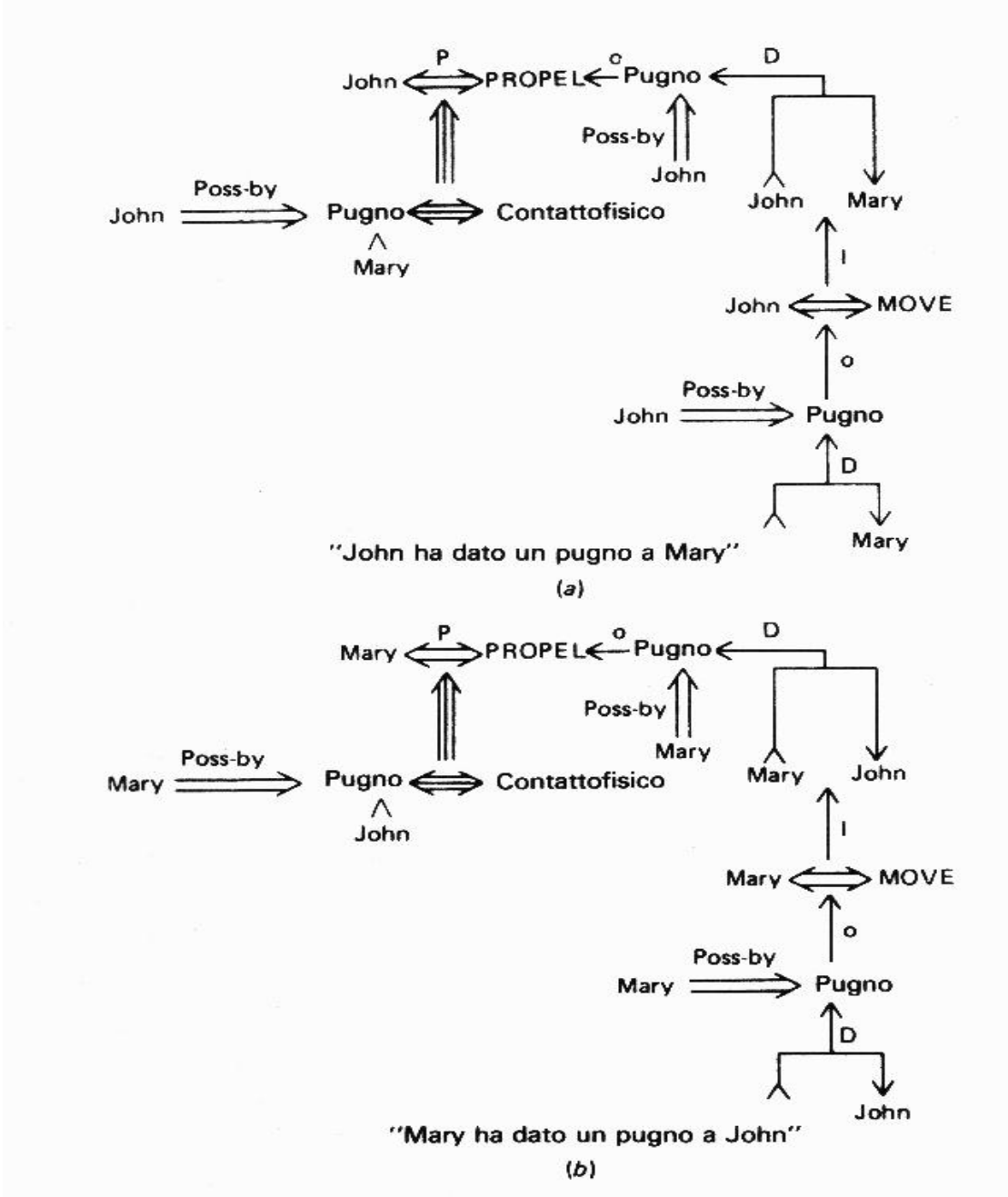
- trasformare eventi complessi in primitive a basso livello comporta molto lavoro spesso **non necessario**

- è richiesto grande spazio di memoria per le rappresentazioni, e di lavoro di immagazzinamento (tutte le volte che si richiama un concetto, occorre istanziarlo).

Esempio: Se si usa la rappresentazione detta dipendenza concettuale e si sa che:

- John ha dato un pugno a Mary
- Mary ha dato un pugno a John

Si hanno le rappresentazioni in memoria (ridondanti):



Non è sempre evidente quali debbano essere le primitive più appropriate, e per certi domini non si può evitare la descrizione a vari livelli.

Esempio: terminologia familiare di Lindsay.

Primitive: padre, madre, figlio, figlia, fratello, sorella.

Per descrivere che Mary è cugina di Sue, come scegliere tra

Mary = figlia(fratello(madre(Sue)))
Mary = figlia(sorella(madre(Sue)))
Mary = figlia(fratello(padre(Sue)))
Mary = figlia(sorella(padre(Sue)))

in modo automatico?
Occorre introdurre anche la primitiva cugina?

Anche se non esiste una risposta definitiva, sembra comunque valido il tentativo di porre il problema come “ricerca di una rappresentazione unica”.

L'accesso alle strutture di conoscenza

Anche per questo problema esistono varie proposte:

- Indicizzare le strutture mediante le parole del contenuto (approccio seguito nella dipendenza concettuale)

Se, ad esempio, si usano gli Script (descrizione di una classe di eventi in termini di contesto, partecipanti e sottoeventi), a ciascuna parola si associa lo script corrispondente.

Come fare con le parole con molti significati?

Esempio: “provare” assume i significati:

- John provò a mettere in moto la macchina (cercò di)
- Sollevando una trave di ferro John provò la sua forza (dimostrò)
- Dopo l’arrosto, John provò la torta di mele (mangiò)
- Quando vide la pistola John provò paura (sentì)
- Entrato nel negozio, John provò la giacca blu (indossò)

- Considerare ogni parola del testo come puntatore a tutte le strutture in cui potrebbe essere contenuta.

  Data una frase, occorre fare l'intersezione per avere la struttura (o strutture) che copre tutte le parole del contenuto.

  Controindicazioni:

- Se è presente una parola leggermente estranea, l'intersezione è vuota
- Il calcolo di tutti gli insiemi di possibilità e poi dell'intersezione è oneroso

- Individuare una parola chiave del testo e selezionare una struttura iniziale. Usare altre parole chiave per raffinare la selezione o per modificare la selezione stessa.

  Controindicazioni:

- Come identificare le parole chiave?
- Come decidere se una parola chiave è o meno più importante di un'altra? La gerarchia delle parole può dipendere dal contesto: in sedia rossa è importante "sedia", ma in luce rossa potrebbe essere importante "rossa"(problema della focalizzazione dell'attenzione).

  Anche qui non esiste una soluzione generale.

Strutture per rappresentare la conoscenza

- Reti semantiche
- Dipendenza concettuale
- Frame
- Script

Sono basate su un'idea comune:

Le entità complesse sono rappresentate come collezione di attributi-valori, cioè hanno una struttura del tipo *slot-and-filler* (caselle da riempire).

Operativamente sono basate su:

- o “memorie associative”, in cui memorizzare terne del tipo oggetto-attributo-valore
- o su “liste di proprietà”

Per aumentare la versatilità inferenziale, oltre agli attributi conviene memorizzare gli attributi inversi. Ad esempio:

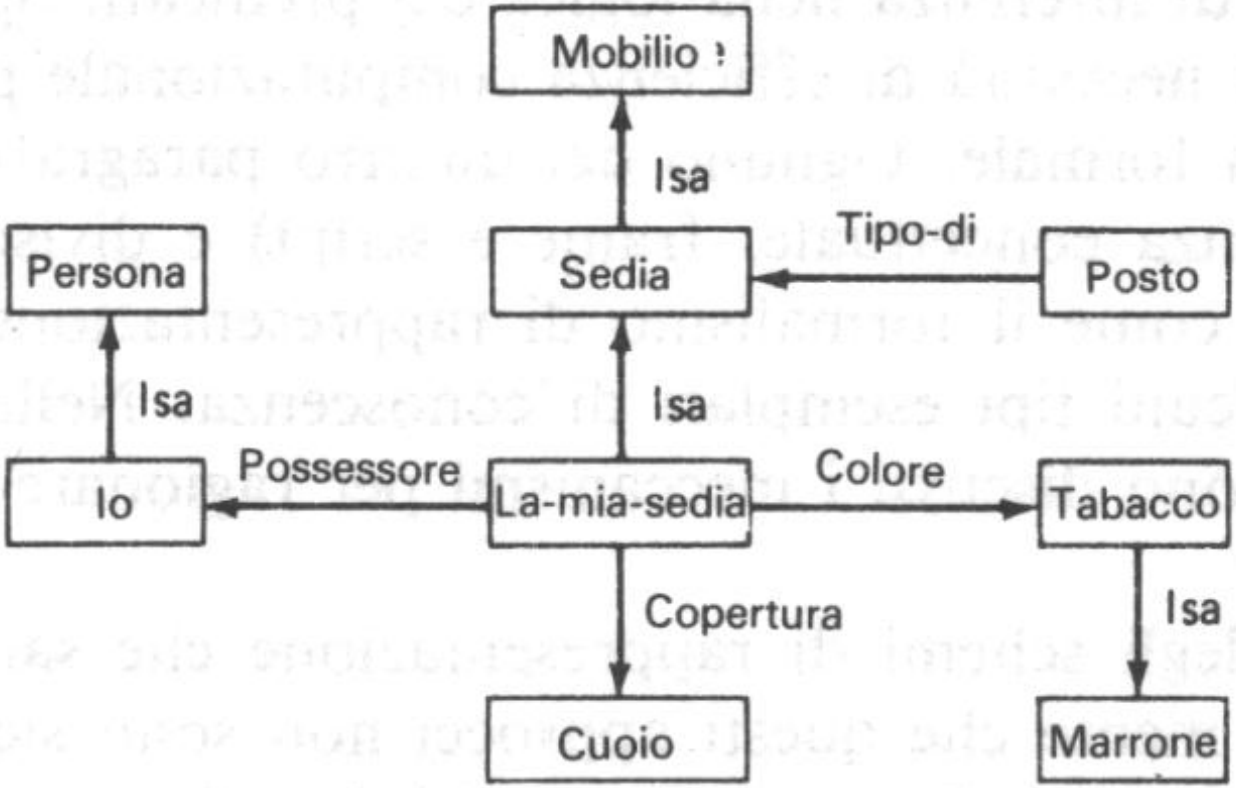
Diretto	Inverso
IS-A	TYPE-OF
FIGLIO	GENITORE
COLORE-DI	ESEMPI-DI-COLORE

La memorizzazione degli inversi deve essere coerente: deve essere un'attività del programma che costruisce la base di conoscenza.

Reti semantiche

- Progettate all'inizio per rappresentare il significato delle parole nel linguaggio naturale (anche come paradigma per psicologi e linguisti)

- L'informazione è rappresentata come insieme di nodi connessi da archi le cui etichette rappresentano le relazioni fra i nodi.



Gli archi sono unidirezionali.

Per esprimere: Io possiedo una sedia occorre aggiungere un nodo (DA-ME) ed un arco (POSSEDUTA).

- Operativamente, sono rappresentate da strutture di memoria basate sulla coppia attributo-valore.

Ad esempio, in LISP, si potrebbero avere le corrispondenze:

nodo → atomo

legami → proprietà

nodi agli estremi → valori

così:

ATOMO	ELENCO PROPRIETÀ
SEDIA	((ISA MOBILIO))
LA-MIA-SEDIA	((ISA SEDIA) (COLORE TABACCO) (COPERTURA CUOIO) (POSSESSORE IO))
IO	((ISA PERSONA))
TABACCO	((ISA MARRONE))
POSTO	((TIPO-DI SEDIA))

Le reti semantiche sono in stretta correlazione con la logica dei predicati. Le relazioni precedenti nella logica dei predicati sarebbero stati (usando predicati a due posti)

ISA (sedia, mobilio)

ISA (io, persona)

COPERTURA (la_mia_sedia, cuoio)

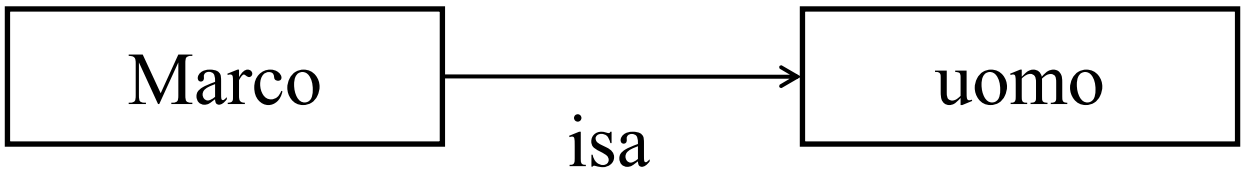
COLORE(la_mia_sedia, tabacco)

Vale anche l'inverso. Ad esempio

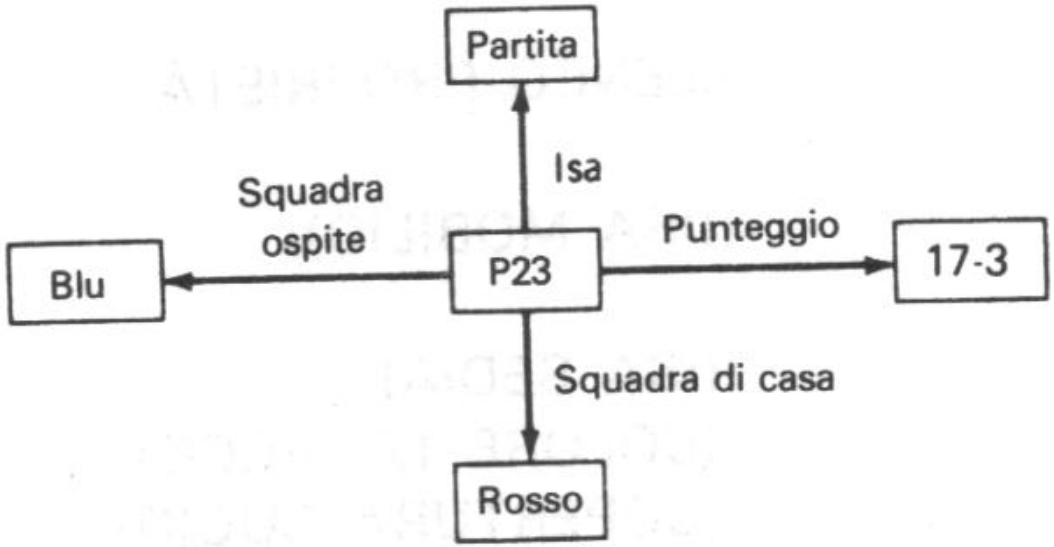
UOMO(Marco)

Può essere riscritto come

ISA (Marco, uomo)

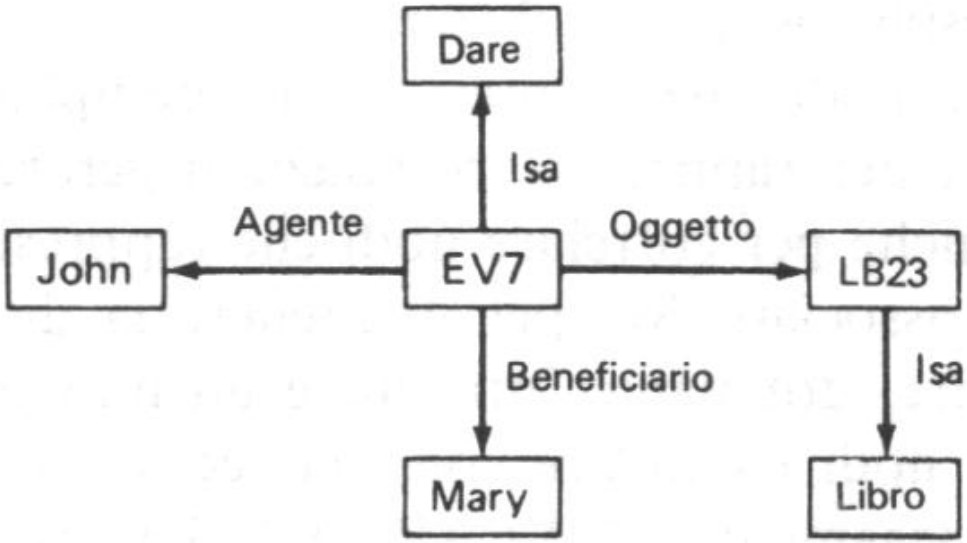


Un fatto relativo ad una partita:
PUNTEGGIO(rosso, blu, (17 3)) diventa:



P23 rappresenta una specifica istanziazione di una partita.

John diede il libro a Mary diventa:



Commento: lo schema rappresenta un'azione in cui c'è un agente, un oggetto, un beneficiario. Questa è una particolare istanziazione (EV7) dell'azione, con una particolare istanziazione dell'oggetto (LB23).

Manipolazione delle reti semantiche

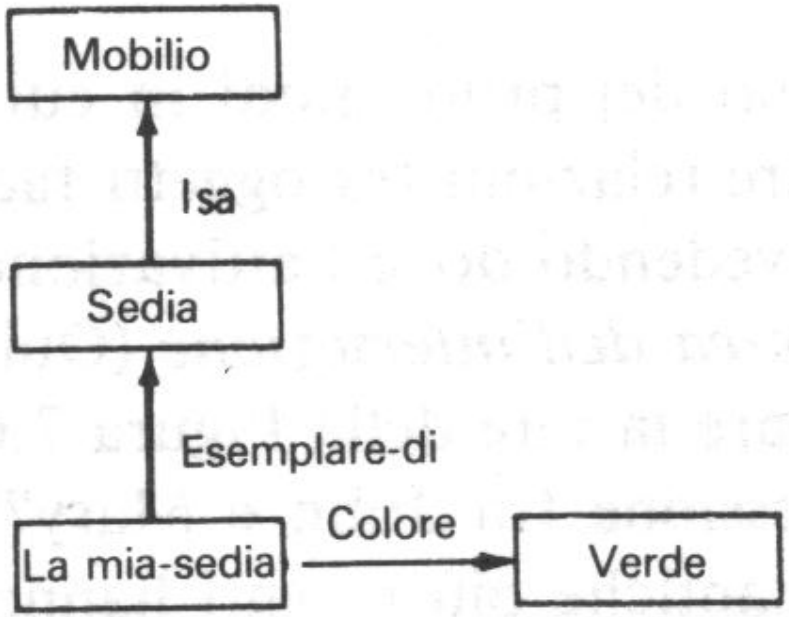
Come rispondere a domande utilizzando reti semantiche ?

Si può usare il metodo detto “ricerca dell’intersezione”: si fa partire un processo di attivazione da due nodi e si vede dove l’attivazione si incontra (nell’esempio precedente, si può vedere “qual è la connessione tra John e Mary”). Questo metodo “mima” i processi cognitivi: è stato abbandonato a favore di metodi più diretti.

Problemi connessi alle rappresentazioni con reti semantiche

- Conviene immagazzinare il concetto nella forma più generale possibile (nodo con legami verso tutte le proprietà di tutti gli esemplari del concetto).

Per tener conto dei casi specifici, si può introdurre un legame del tipo ESEMPLARE-DI, così



Nota: Se ho soltanto il nodo

Non posso rappresentare un fatto nuovo come:

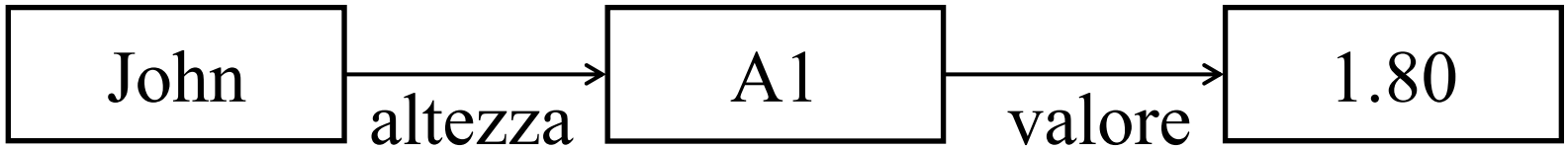
La sedia di Mary è verde

- Conviene tenere separate le proprietà e le relazioni per quanto possibile.

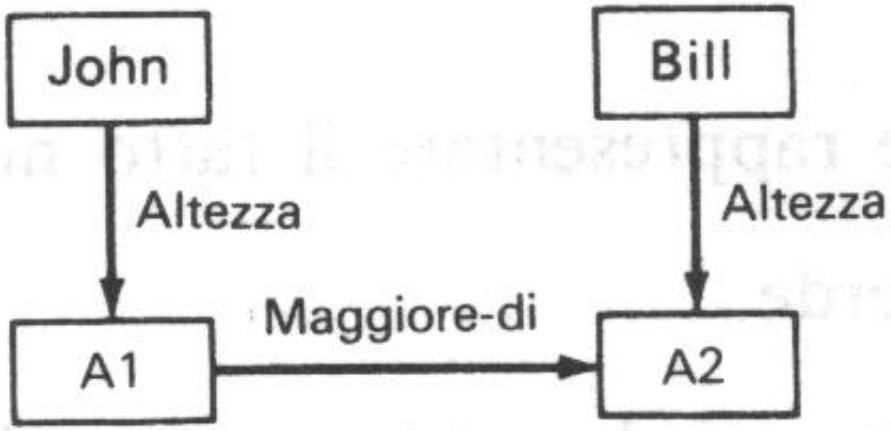
Esempio

Invece di

Conviene porre:



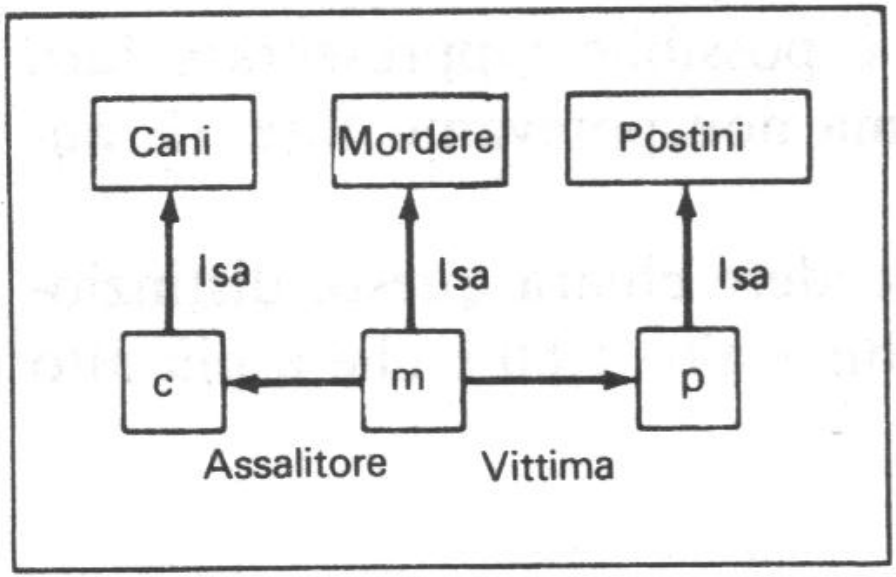
che rende possibile evidenziare una relazione come



- Per risolvere il problema della quantificazione, conviene partizionare la rete semantica in un insieme gerarchico di spazi, ognuno dei quali corrisponde al campo di azione di una o più variabili.

Esempio:

- Singolo fatto: *Il cane ha morso il postino*
  Rappresentazione:

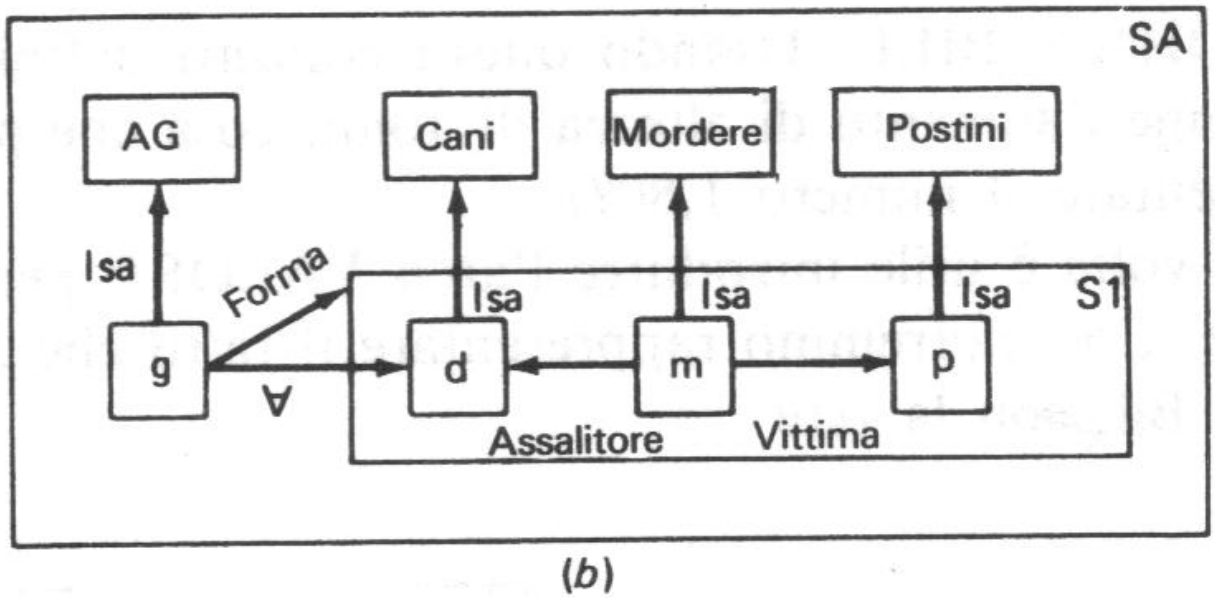


(*a*)

- Fatto universalmente ed esistenzialmente quantificato: *Ogni cane ha morso un postino*

$$(\forall x\ cane(x) \rightarrow (\exists y\ postino(y) \wedge mordere(x, y)))$$

Rappresentazione:



(*b*)

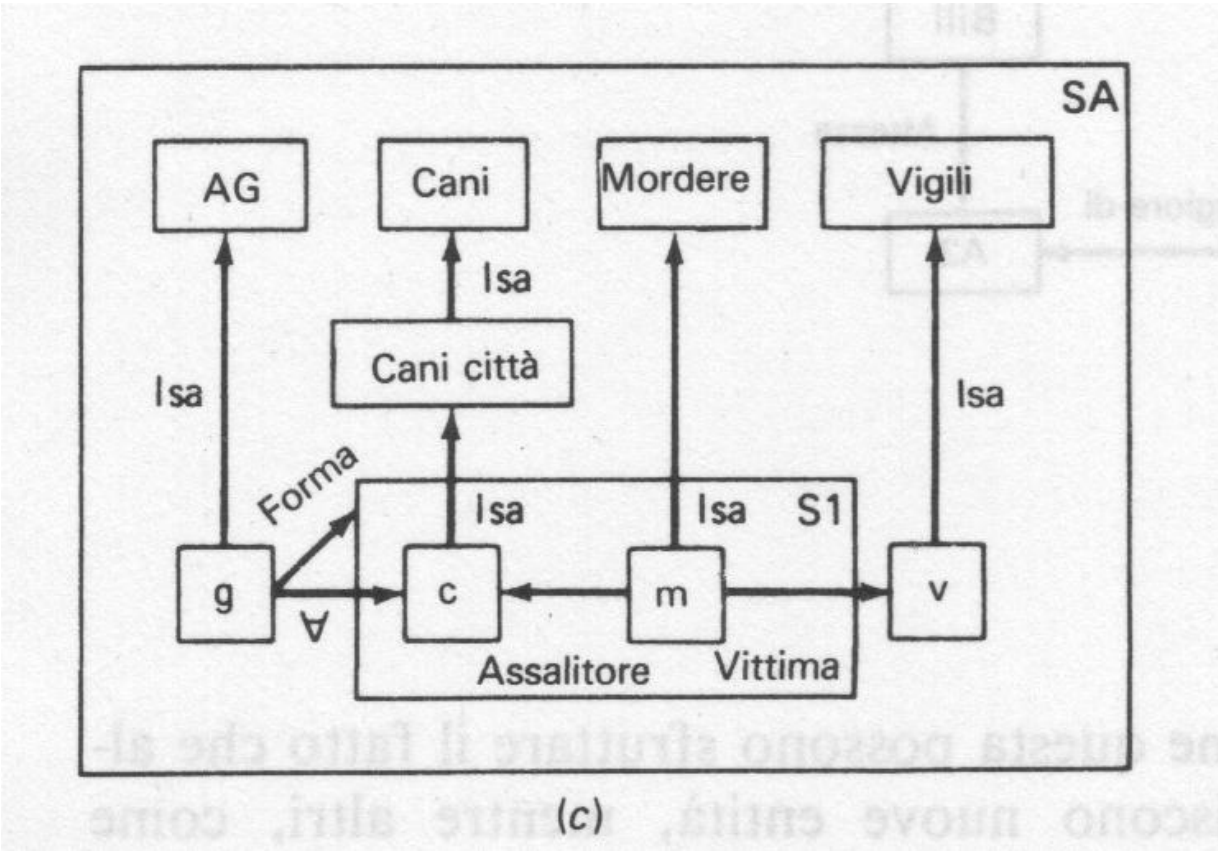
g è un esemplare della classe AG (asserzione generale sul mondo, nelle quali c'è una o più quantificazioni universali).

Ogni elemento di AG ha 2 attributi:

- una FORMA (cioè la relazione asserita)

- una o più connessioni , che punta alla (o alle) variabile quantificata universalmente (nell'esempio a *c*, elemento della classe cani). Si noti che *m* è *p* sono comprese nello "spazio" della variabile quantificata universalmente: per ogni *c*, esiste un'azione di mordere *m* che ha come vittima un *p* (postino).

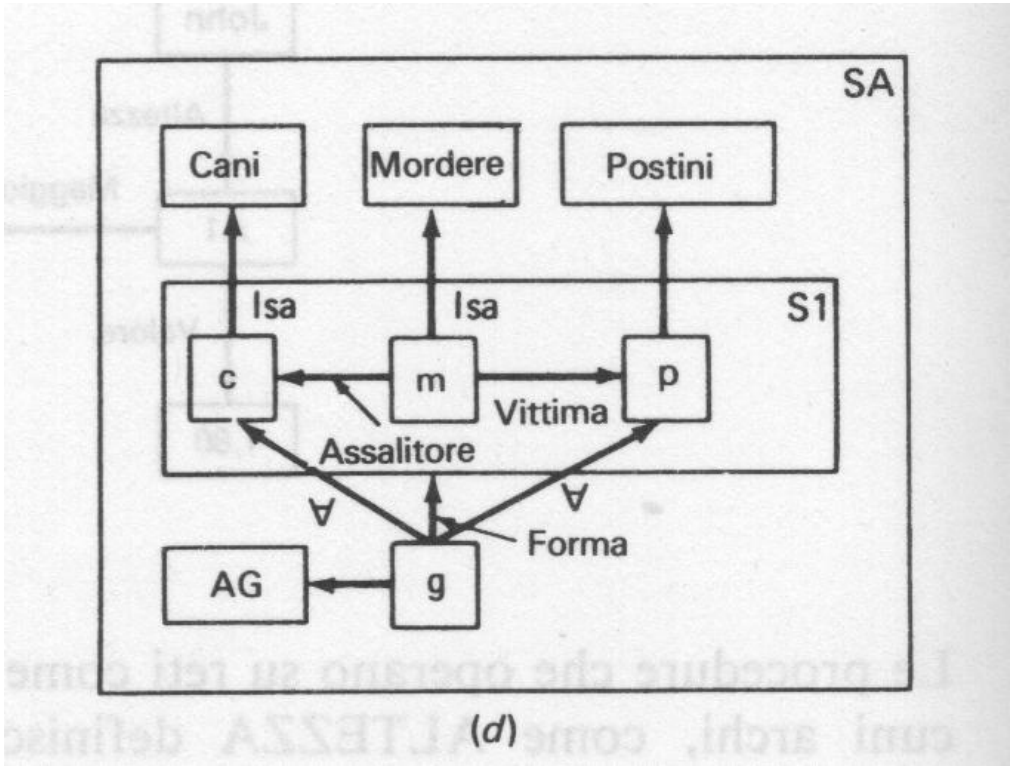
- Si comprende meglio come opera lo spazio di partizionamento se si rappresenta la frase: *ogni cane della città ha morso un vigile*



v è il posto fuori dallo spazio in cui opera la quantificazione universale: tutti i cani mordono lo stesso postino

Per rappresentare la frase: *ogni cane ha morso ogni postino*

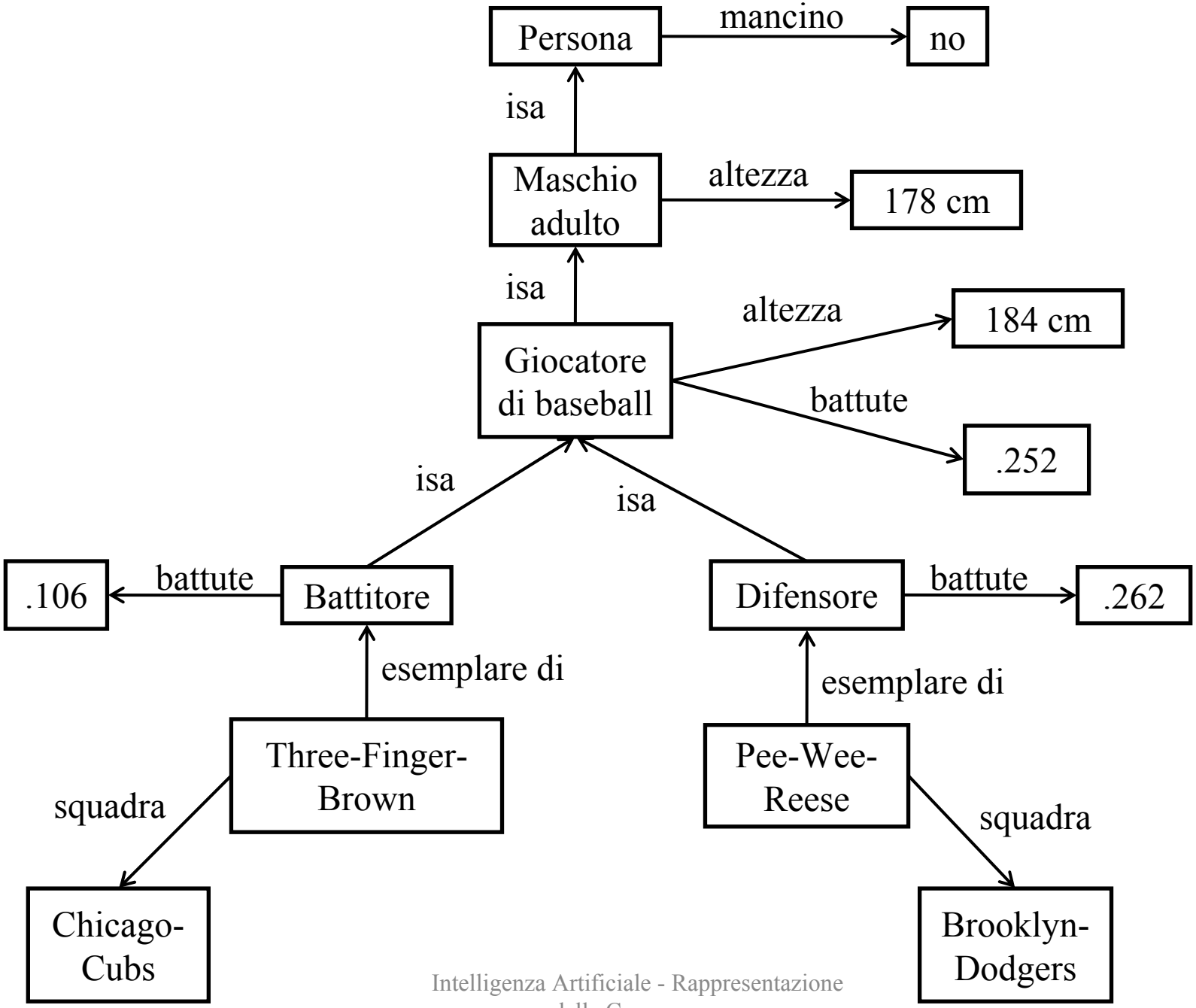
si crea una rappresentazione con due legami , così:



(*d*)

Algoritmi per l'ereditarietà

Si consideri la seguente conoscenza ereditabile:


Persona
mancino
no
isa
Maschio adulto
altezza
178 cm
isa
Giocatore di baseball
altezza
184 cm
battute
.252
isa
isa
.106
battute
Battitore
Difensore
battute
.262
esemplare di
esemplare di
Three-Finger-Brown
Pee-Wee-Reese
squadra
squadra
Chicago-Cubs
Brooklyn-Dodgers

Algoritmo (di principio) dell'ereditarietà:

Per ottenere il valore *V* dell'attributo *A* di un esemplare *O*:

I. Si trova *O* nella base di conoscenza.

II. Se c'è un valore per l'attributo *A*, si riporta tale valore.

III. Altrimenti, si vede se è presente un valore per l'attributo *esemplare-di*: se non c'è, si termina con fallimento.

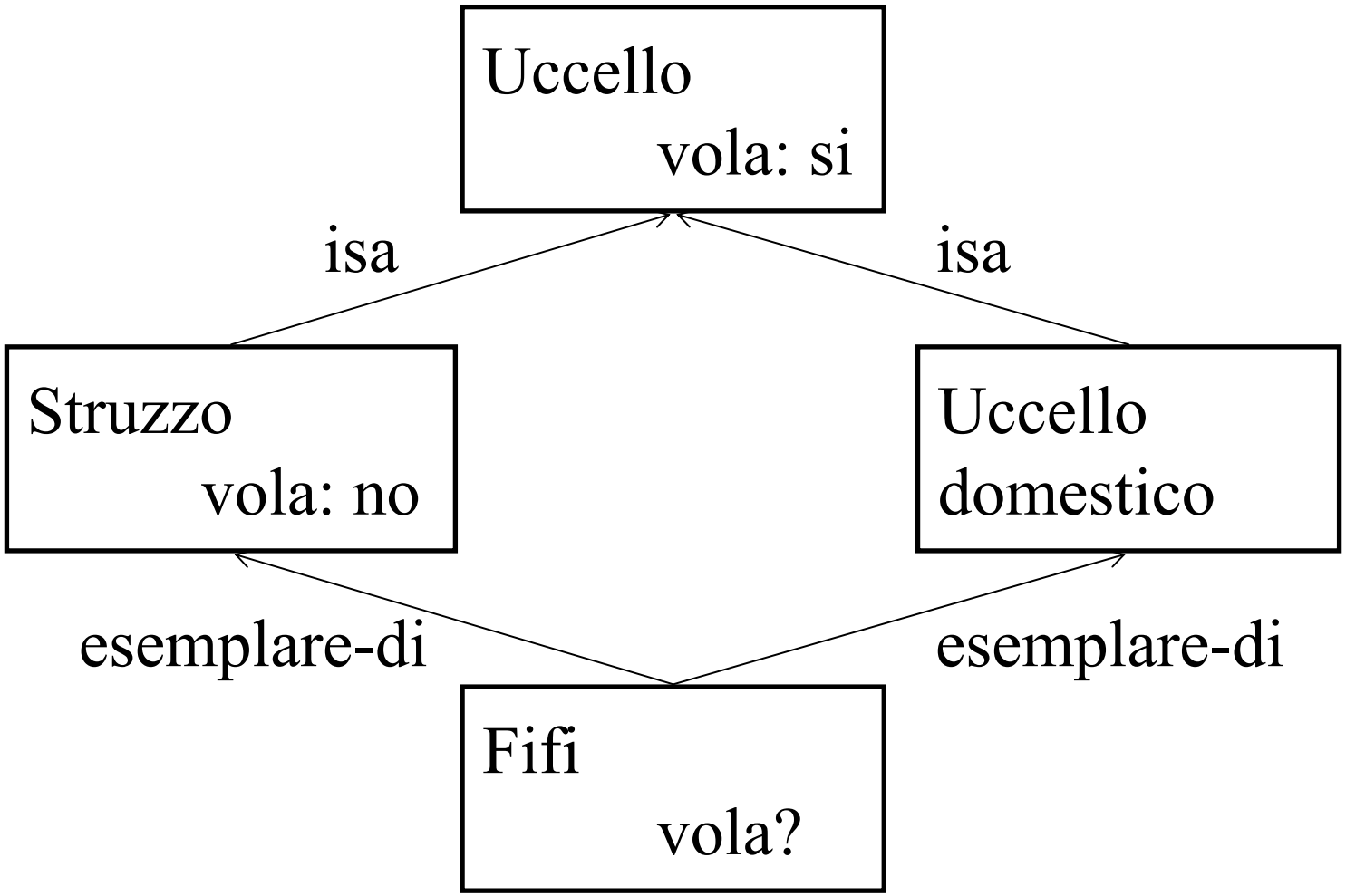
III. Altrimenti, ci si sposta sul nodo corrispondente al valore dell'attributo *esemplare-di*, e si cerca un valore per l'attributo. Se c'è, si riporta il valore trovato.

IV. Altrimenti si ripetono le seguenti operazioni finché o non c'è nessun valore per l'attributo *isa* o si determina una risposta:

A. ci si sposta sul nodo corrispondente al valore dell'attributo *isa*.
B. si vede se è presente un valore per l'attributo *A*, e in caso affermativo si riporta tale valore.

Limiti di questo algoritmo: cosa fare quando c'è più di un valore per l'attributo *esemplare-di* o per l'attributo *isa*?

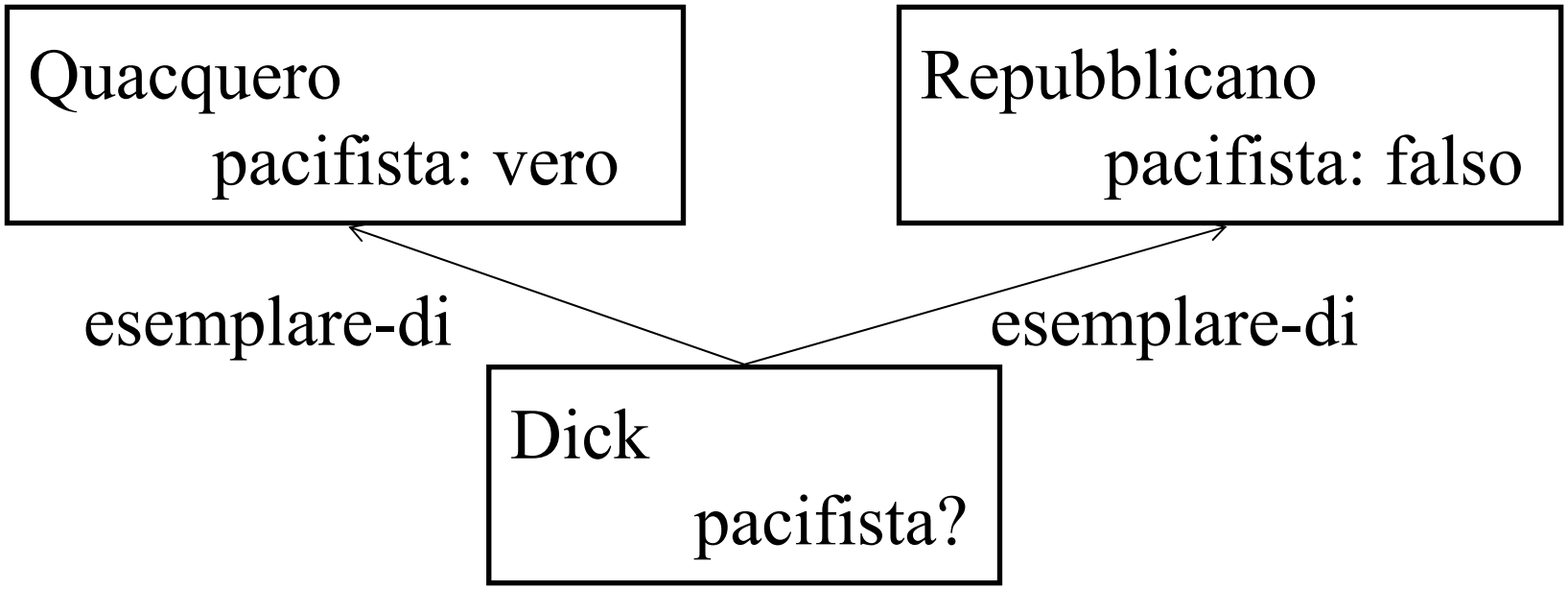
Primo esempio:



Il cammino di destra (Fifi → Uccello domestico → Uccello) permette di derivare una risposta positiva in contrasto con quella negativa derivata dal percorso di sinistra (Fifi → Struzzo).

Manca nell'algoritmo un meccanismo per attraversare la gerarchia *isa* che garantisca che la conoscenza specifica prevalga sempre sui fatti più generali.

Secondo esempio:



Per derivare una risposta si attraversano archi *esemplare-di* multipli, e attraverso questi cammini è possibile trovare più di una risposta.

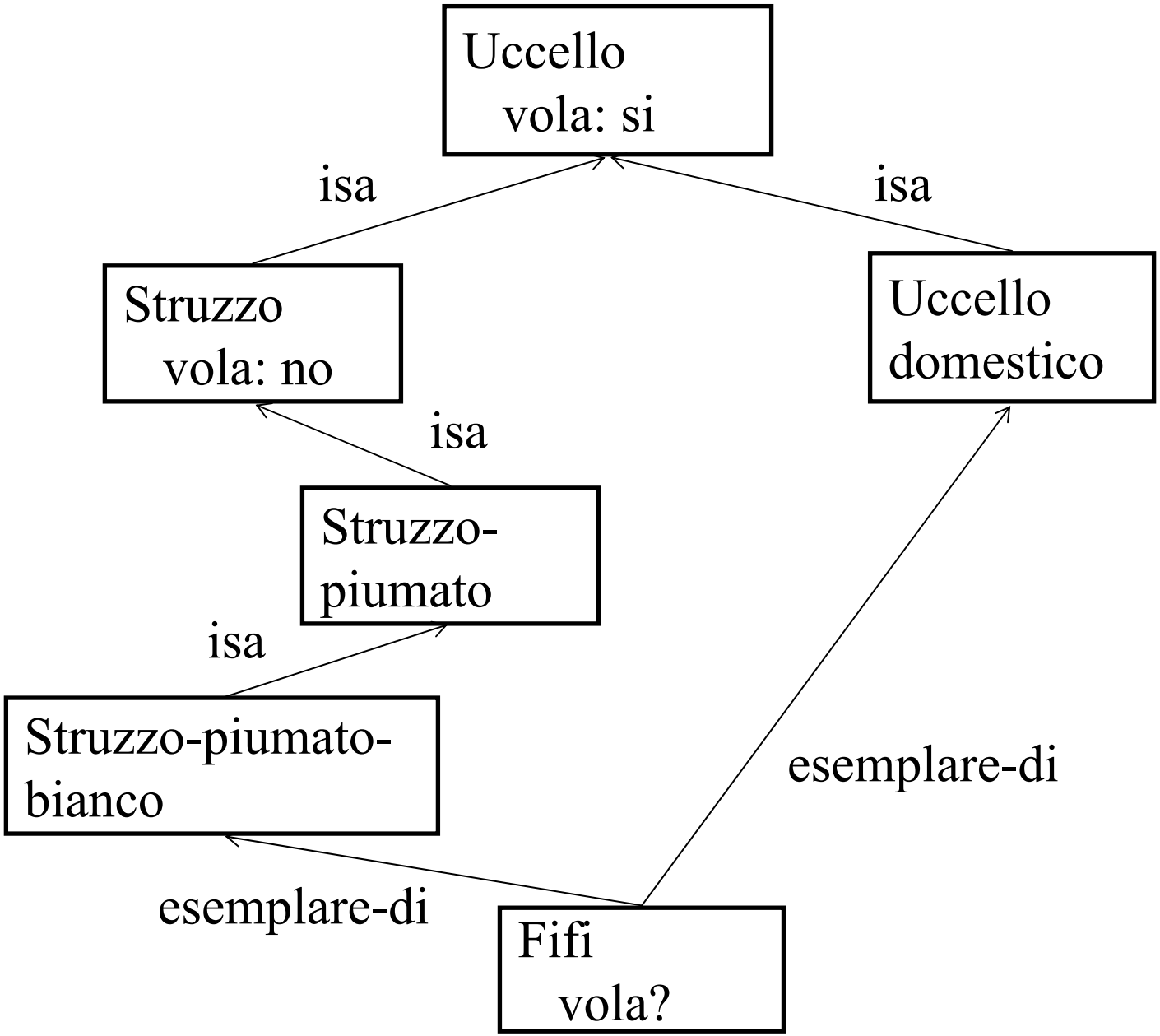
L'algoritmo non riesce a rilevare l'intrinseca ambiguità di tale rappresentazione.

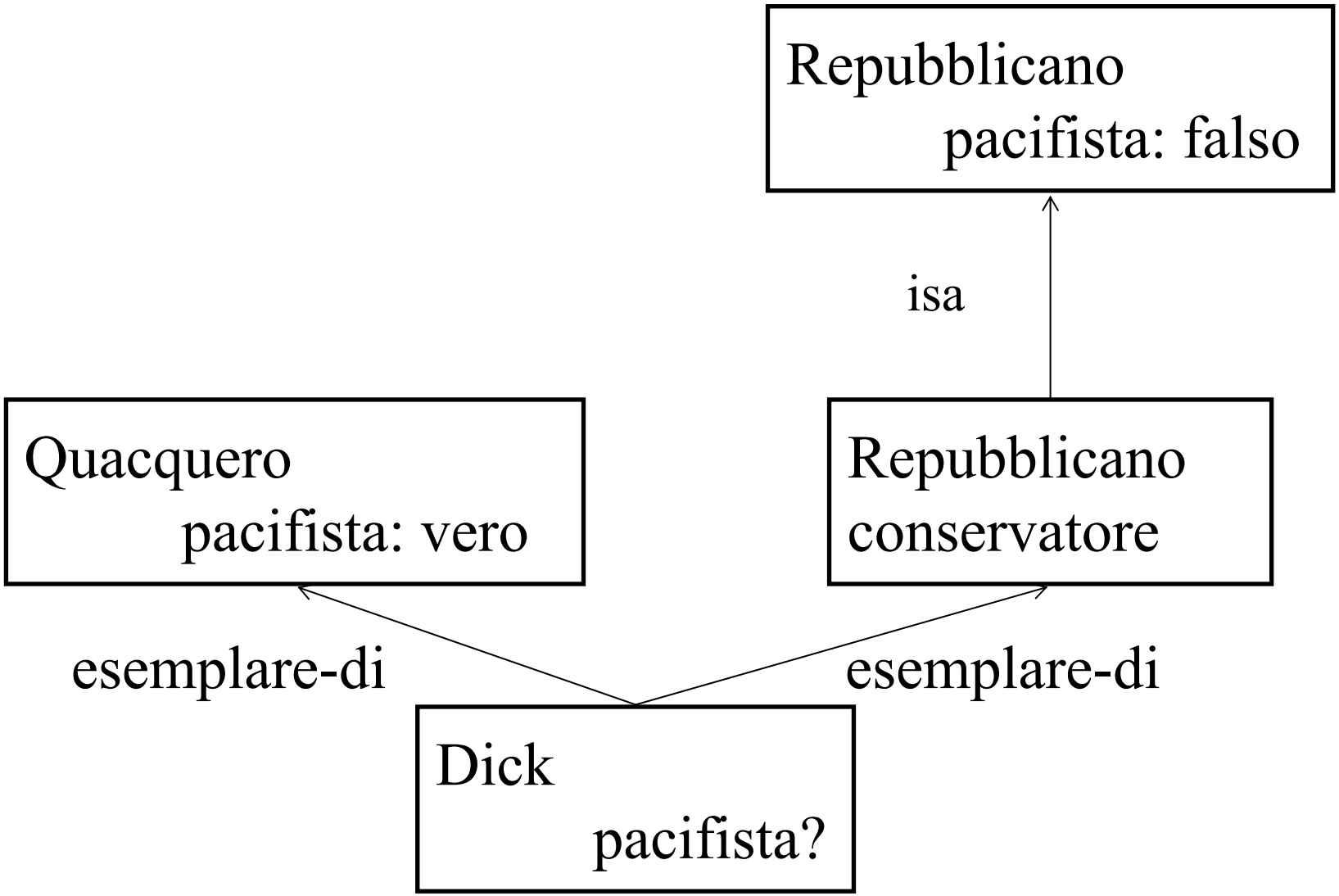
Si potrebbe pensare di risolvere il problema inserendo nell'algoritmo un controllo relativo alla lunghezza dei cammini:

si esegue una ricerca in ampiezza a partire dall'elemento per il quale c'è bisogno del valore di una casella, percorrendo i suoi archi *esemplare-di* e quindi i suoi archi *isa* sempre all'insù, e terminando non appena un cammino produce un valore.

Funzionerebbe nel primo esempio, fornendo la risposta negativa corretta, e anche nel secondo, rilevando l'ambiguità mediante due risposte di uguale livello tra loro contraddittorie.

Non funzionerebbe nei seguenti due esempi:


Uccello
vola: si
isa
isa
Struzzo
vola: no
Uccello
domestico
isa
Struzzo-
piumato
isa
Struzzo-piumato-
bianco
esemplare-di
esemplare-di
Fifi
vola?


Repubblicano
pacifista: falso
isa
Quacquero
pacifista: vero
Repubblicano
conservatore
esemplare-di
esemplare-di
Dick
pacifista?

Il problema è che la lunghezza dei cammini non sempre corrisponde al livello di generalità delle classi, ma piuttosto al grado di elaborazione di una classe in una base di conoscenza.

Touretzky (1986) ha introdotto la nozione di *distanza inferenziale* così definita:

"*Classe1* è più vicina a *Classe2* di *Classe3* se e solo se *Classe1* ha un cammino di inferenza che termina in *Classe3* passando per *Classe2* (in altre parole, *Classe2* è tra *Classe1* e *Classe3*)".

Si può allora definire il risultato dell'eredità come: l'insieme dei valori in competizione per una casella *S* in un frame *F* contiene tutti quei valori che

- possono essere derivati da qualche frame *X* che sta sopra *F* nella gerarchia *isa*;

- non sono contraddetti da qualche frame *Y* la cui distanza inferenziale da *F* sia più corta di quella esistente tra *X* e *F*.

Si noti che, secondo la definizione, valori in competizione che derivano da frame non confrontabili, continuano ad essere in competizione.

Con questa modifica l'algoritmo opera correttamente negli ultimi due esempi.

Poiché *Struzzo* è una sottoclasse di *Uccello*, sarà più vicino di *Uccello* a tutti i propri esemplari, indipendentemente da quante altre classi vengano aggiunte al sistema.

Nell'altro esempio, si ottengono due risposte, nessuna delle quali è più vicina a *Dick* dell'altra.

Algoritmo generale di eredità delle proprietà:

Per determinare un valore *V* per una casella *S* di un esemplare *F*, si esegue:

I. Si pone *CANDIDATI* a vuoto.

II. Si esegue una ricerca in ampiezza o in profondità a partire da *F*, verso l'alto nella gerarchia *isa*, seguendo tutti gli archi *esemplare-di* e *isa*. Ad ogni passo, si guarda se c'è memorizzato un valore per *S* o per una delle sue generalizzazioni:

A. se si trova un valore, lo si aggiunge a *CANDIDATI* e si termina quel ramo della ricerca;
B. se non si trova alcun valore ma ci sono archi *esemplare-di* e *isa* verso l'alto, si seguono;
C. altrimenti si termina il ramo.

III. Per ogni elemento *C* di *CANDIDATI*:

A. si guarda se c'è qualche altro elemento di *CANDIDATI* che è derivato da una classe più vicina ad *F* della classe dalla quale proviene *C*;
B. se c'è, si rimuove *C* da *CANDIDATI*.

IV. Si controlla la cardinalità di *CANDIDATI*:

A. se è 0 si segnala che non è stato trovato alcun valore;
B. se è 1 si riporta come *V* l'unico elemento di *CANDIDATI*;
C. se è maggiore di 1 si riporta una contraddizione.

L'algoritmo certamente termina in quanto la gerarchia *isa* è rappresentata da un grafo aciclico.

La dipendenza concettuale

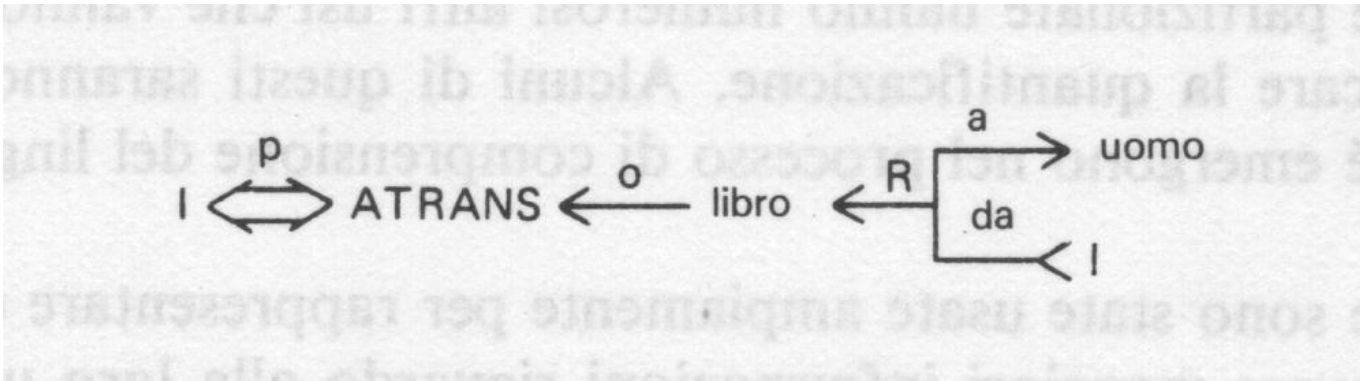
- Più che rappresentare le caratteristiche statiche del dominio, è intesa a modellare il processo dinamico di comprensione di una frase, e quindi a facilitare le inferenze.
- È indipendente dalla lingua in cui si esprimono i concetti
- La sintassi e la semantica sono integrate (cioè la grammatica della lingua non è rappresentata esplicitamente).
- È basata su un limitato numero di Azioni Primitive mediante le quali esprimere tutti gli eventi (15 azioni).

Ad ognuna delle azioni è associabile una procedura di inferenza che guida il processo di comprensione e ha la funzione di “estrarre” dal contesto le informazioni implicite nella frase analizzata.

Esempio: la frase

Mary diede un libro a John

è rappresentata così:



dove i simboli hanno il seguente significato:

- Le frecce indicano la direzione di dipendenza
- La doppia freccia indica il doppio legame fra l'attore e l'azione
- *p* indica il tempo passato
- ATRANS è una delle azioni primitive usate dalla teoria. Indica il passaggio di possesso
- *o* indica la relazione di oggetto
- *R* indica la relazione di ricevente

Le azioni primitive sono:

ATRANS trasferimento di una relazione astratta (per esempio, dare)

PTRANS trasferimento della posizione fisica di un oggetto (per esempio, andare)

PROPEL applicazione di forza fisica ad un oggetto (per esempio, spingere)

MOVE movimento di una parte del corpo da parte del suo possessore (per esempio, dare un calcio)

GRASP l'atto di afferrare un oggetto da parte di un essere animato (per esempio, mangiare)

EXPEL espulsione di qualcosa dal corpo di un essere animato (per esempio, piangere)

MTRANS trasferimento di informazione mentale (per esempio, raccontare)

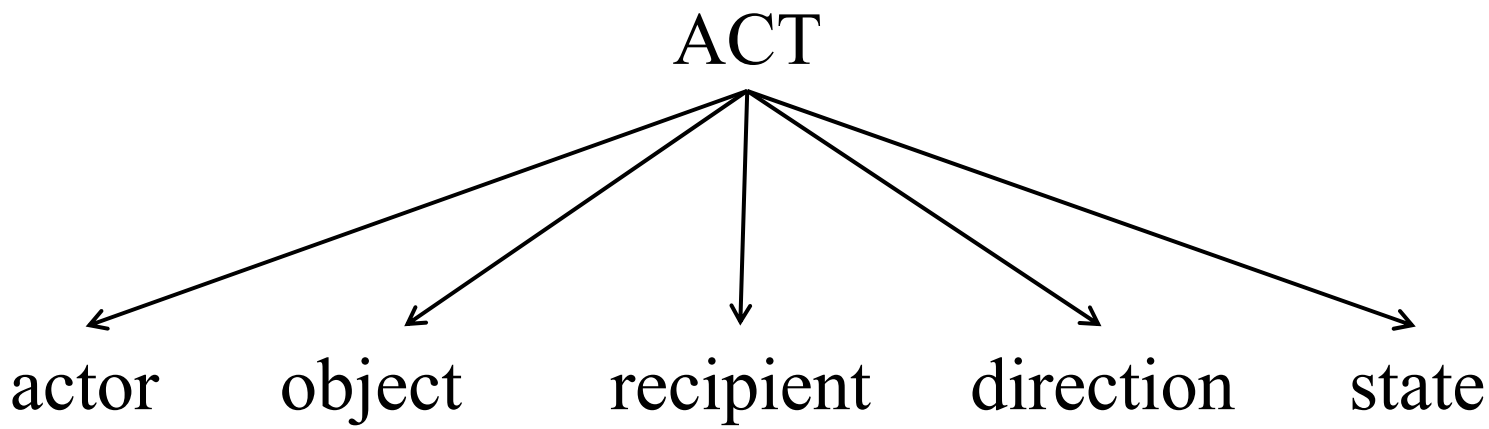
MBUILD costruzione di una nuova informazione a partire da quella precedente (per esempio, decidere)

SPEAK produzione di suoni (per esempio, dire)

ATTEND focalizzazione di un organo di senso verso uno stimolo (per esempio, ascoltare), chiamata CONC in alcuni dei primi lavori di Schank (Schank, 1973b)

Costituiscono i blocchi elementari mediante i quali si possono costruire azioni a più alto livello corrispondenti alle frasi: sono quindi i nuclei intorno ai quali è incentrata la concettualizzazione.

Una *concettualizzazione* è la descrizione di un evento mediante un'azione primitiva più un insieme di *casi* ad essa collegati.



I costituenti indicati dai *casi* possono essere:

PP (Picture Producer)	Concetti nominali
ACT (Action)	Azioni primitive
LOC (Location)	Luoghi
T (Time)	Tempi

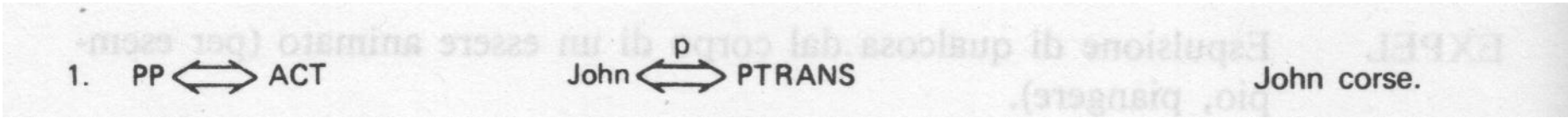
AA (Action Aider) — Descrittori supplementari di azioni (es. "velocità" per un movimento)

PA (Picture Aider) — Descrittori supplementari di entità (es. "altezza" di una persona).

A partire da queste categorie concettuali primitive si possono costruire strutture di dipendenza.

Le dipendenze fra concettualizzazione corrispondono a (e servono ad esprimere) relazioni semantiche fra i concetto sottostanti.

Ecco degli esempi:



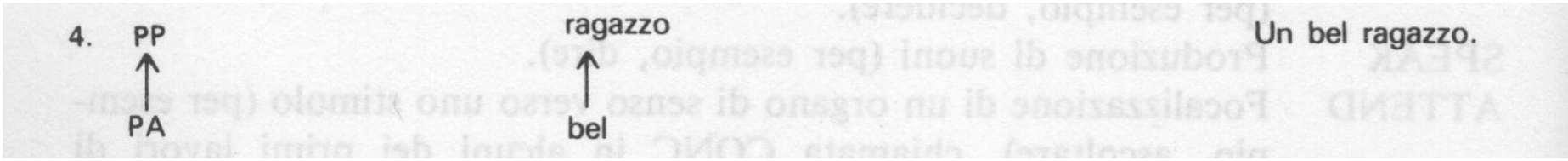
La regola 1 descrive la relazione fra un attore e l'evento da lui causato. Questa è una dipendenza a due sensi, poiché né l'attore né l'evento possono essere considerati primari. La lettera *p* sopra il legame di dipendenza indica il tempo passato.

2. PP ⟺ PA John ⟺ altezza (> media) John è alto.

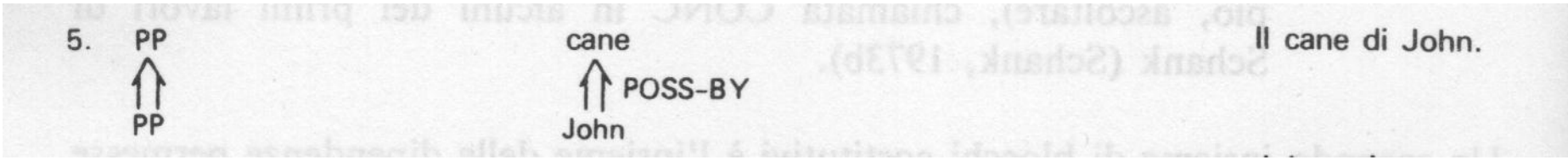
La regola 2 descrive la relazione fra un PP e un PA che si asserisce lo descriva. Molte descrizioni di stato, come l'altezza, sono rappresentate nella CD con scale numeriche.

3. PP ⟺ PP John ⟺ dottore John è un dottore.

La regola 3 descrive la relazione fra due PP, uno dei quali appartiene all'insieme definito dall'altro.

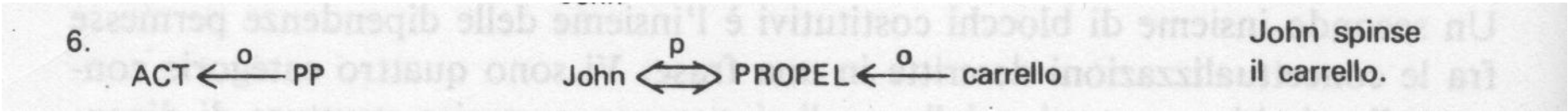


La regola 4 descrive la relazione fra un PP e un attributo che è già stata predicato su esso. La direzione della freccia è verso il PP descritto.

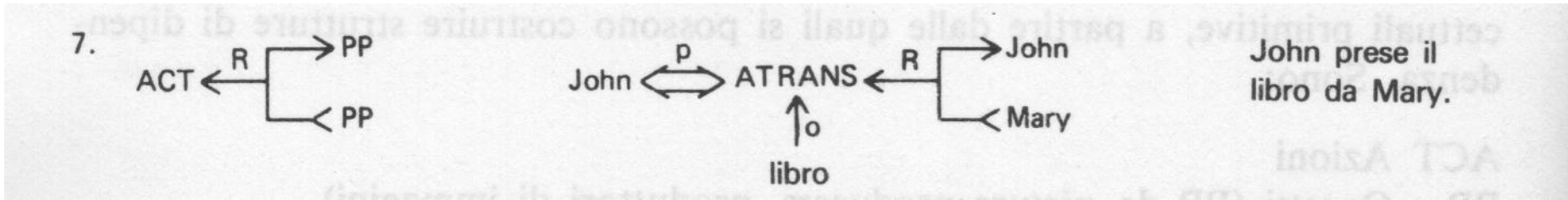


La regola 5 descrive la relazione fra due PP, uno dei quali fornisce un tipo particolare di informazione sull'altro.

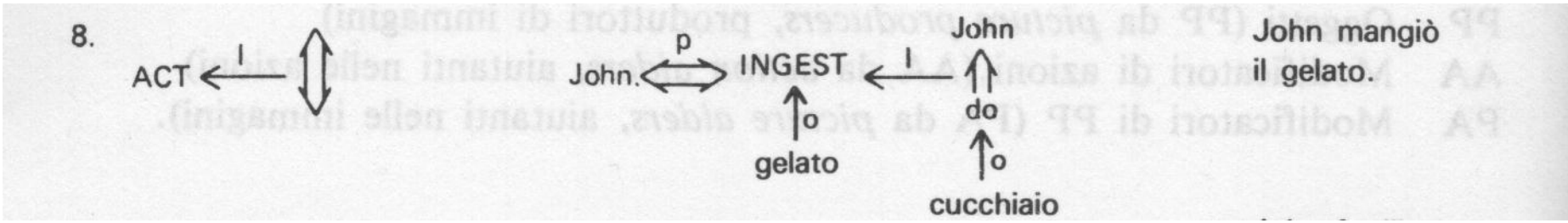
I tre tipi più comuni di informazione da fornire in questo modo sono il possesso (mostrato come POSS-BY), posizione (mostrata come LOC), e il contenimento in senso fisico (mostrato come CONT). La direzione della freccia è nuovamente verso il concetto descritto.



La regola 6 descrive la relazione fra un ACT e il PP che costituisce l'oggetto di quell'atto. La direzione della freccia è verso l'ACT poiché il contesto dello specifico ACT determina il significato della relazione dell'oggetto.

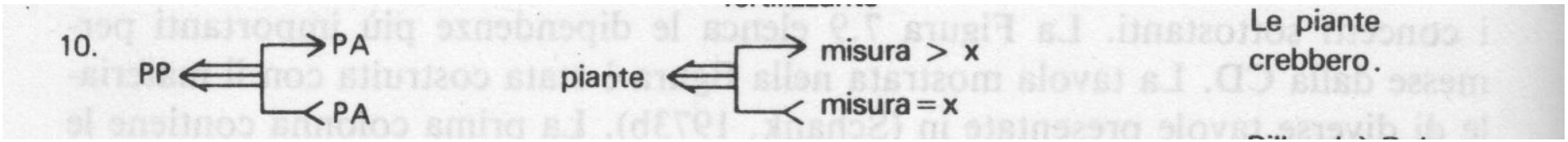


La regola 7 descrive la relazione fra un ACT e la fonte e il ricevente dell'ACT.

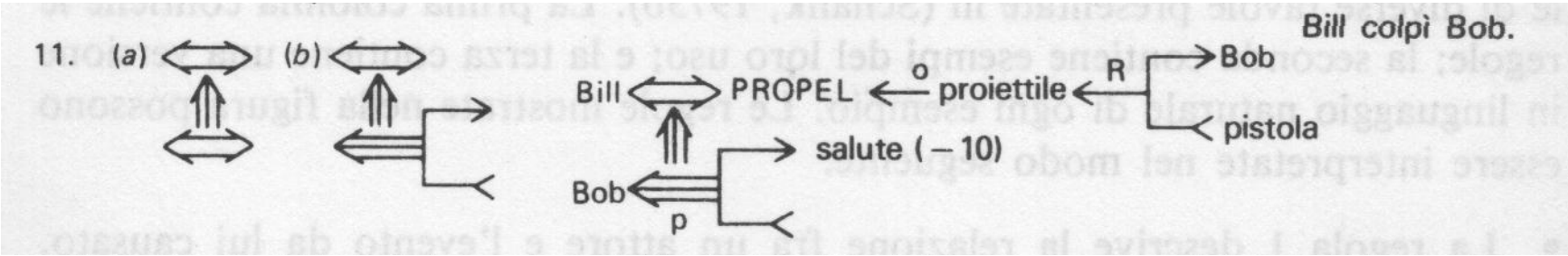


La regola 8 descrive la relazione fra un ACT e lo strumento con cui è attuato. Lo strumento deve sempre essere una concettualizzazione completa (vale a dire, deve contenere un ACT), e non semplicemente un singolo oggetto fisico.

La regola 9 descrive la relazione fra un ACT e la sua fonte e destinazione fisiche.

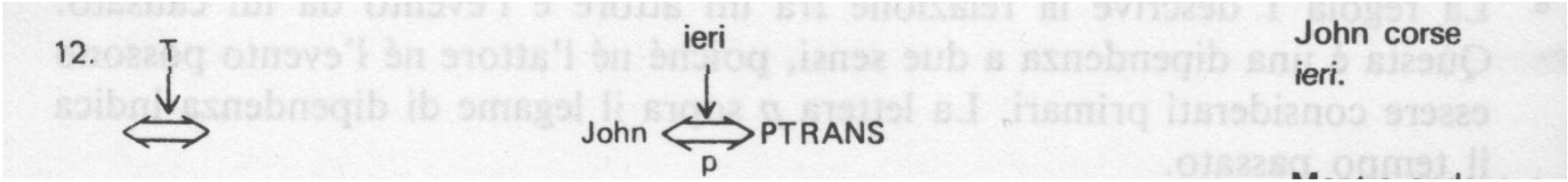


La regola 10 rappresenta la relazione fra un PP e uno stato in cui ha avuto inizio e un altro in cui è terminato.

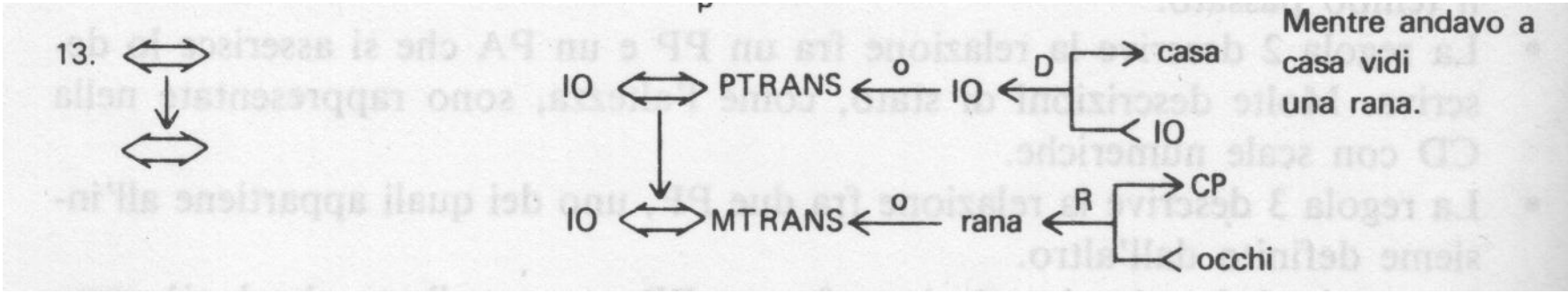


(si noti la freccia a 3 linee, che indica causalità)

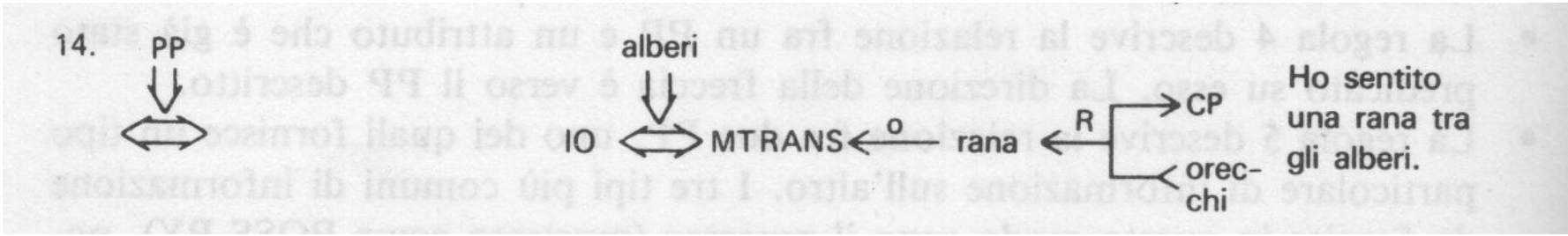
La regola 11 descrive la relazione fra una concettualizzazione e un'altra che la causa. Si noti che le frecce indicano la dipendenza di una concettualizzazione da un'altra e quindi puntano nella direzione opposta rispetto alle frecce di implicazione. Le due forme della regola descrivono la causa di un'azione e la causa di un cambiamento di stato.



La regola 12 descrive la relazione fra una concettualizzazione e il tempo in cui l'evento che descrive si è verificato.



La regola 13 descrive la relazione fra una concettualizzazione e un'altra che è il tempo della prima. L'esempio di questa regola mostra anche come la CD sfrutti un modello del sistema di elaborazione dell'informazione umano; *vedere* è rappresentato come il trasferimento di informazione fra gli occhi e l'elaboratore conscio.



La regola 14 descrive la relazione fra una concettualizzazione e il luogo in cui è avvenuta.

Sono previsti dei modificatori per tener conto, in una concettualizzazione, del tempo, del modo o del tipo di forma verbale:

p passato

f futuro

t transizione

t_s inizio della transizione

t_f fine della transizione

k in corso

? interrogativo

/ negativo

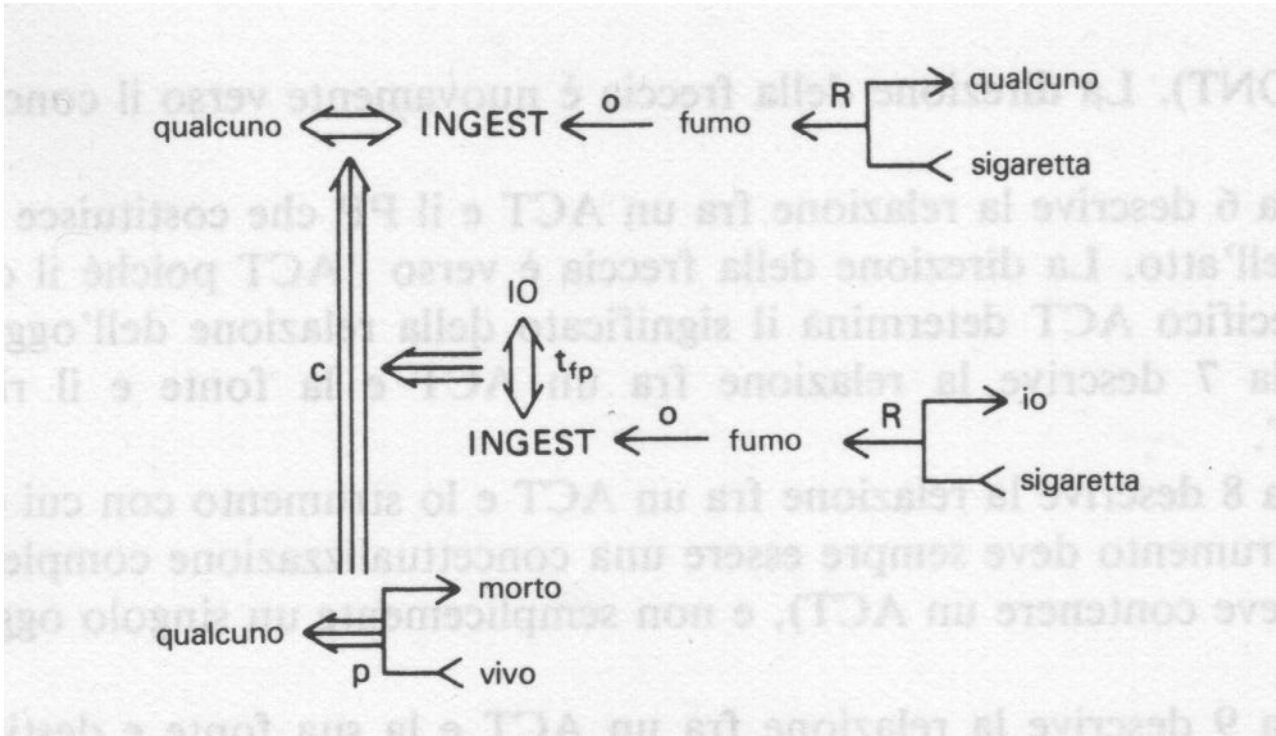
nil presente

delta senza tempo

c condizionale

Esempio:

Poiché fumare può uccidere, io ho smesso



NB: : evento finito nel passato

Vantaggi della dipendenza concettuale

1. Sono necessarie meno regole di inferenza di quelle che sarebbero richieste se la conoscenza non fosse fatta risalire a primitive
2. Molte inferenze sono già contenute nella rappresentazione stessa
3. La struttura iniziale costruita per rappresentare l'informazione contenuta in una frase avrà dei buchi che devono essere riempiti. Questi buchi possono servire per focalizzare l'attenzione del programma che deve capire frasi che seguono.

Il primo punto si giustifica col fatto che non occorre associare ad ogni parola una regole d'inferenza: è sufficiente associarla all'ACT primitivo che descrive parole che esprimono lo stesso concetto.

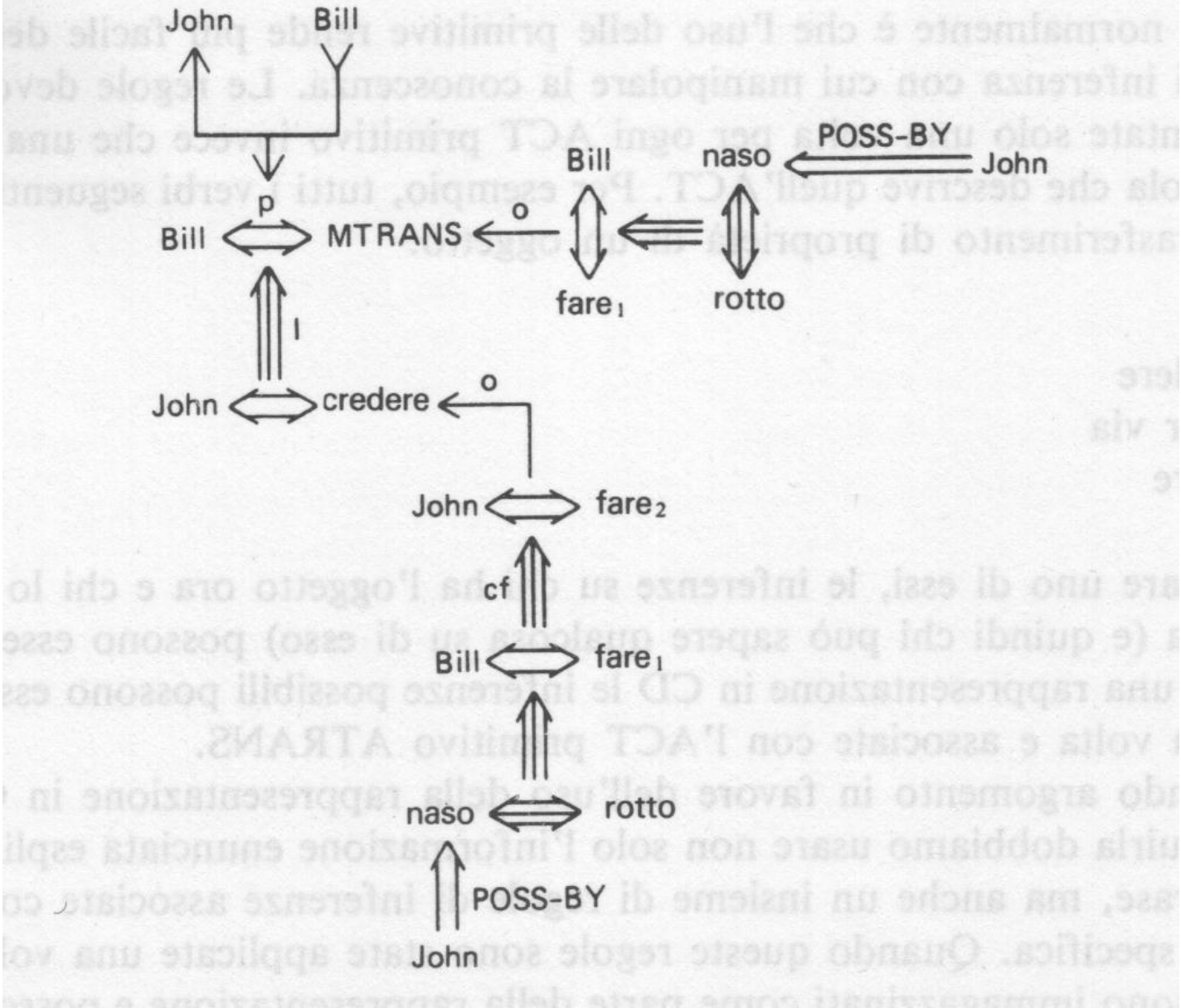
Ad esempio, dare, prendere, portar via, donare si possono riferire tutte ad un ACT che descrive il trasferimento di proprietà di un oggetto

Il secondo punto è insito nel tipo di rappresentazione stessa: per ottenere questa, non si usa solo l'informazione enunciata esplicitamente nella frase, ma anche i concetti ad essa associata (e quindi le connessioni tra concetti che sottostanno alle regole di inferenza)

Una frase come

Bill minacciò John di spaccargli il naso

può essere rappresentata come:


John
Bill
p
Bill
MTRANS
o
Bill
naso
POSS-BY
John
fare1
rotto
l
John
credere
o
John
fare2
cf
Bill
fare1
naso
rotto
POSS-BY
John

Il terzo punto è ancora illustrato nell'esempio precedente. La presenza dell'azione fittizia $fare_2$ fornisce una indicazione su quali altri eventi o oggetti sono importanti per la comprensione della frase.

Essa funge da stimolo a cercare di riempire i buchi vuoti (processi attentivi ovvero focalizzazione dell'attenzione).

Svantaggi della dipendenza concettuale

- Le primitive sono tutte a basso livello, e possono comportare inefficienza nel loro uso.

- Permette di rappresentare solo eventi e quindi non esaurisce tutte le necessità di rappresentazione della conoscenza (ad esempio, gli oggetti possono essere presenti solo come atomi).

Frame

- Sono strutture di rappresentazione a caselle da riempire, adatte per situazioni complesse

- Idea di base (Minsky): quando si affrontano nuove situazioni si fa uso di conoscenze preesistenti (immagazzinate in forma di schemi)

- Un frame descrive una classe di oggetti (SEDIA, STANZA, ecc.)

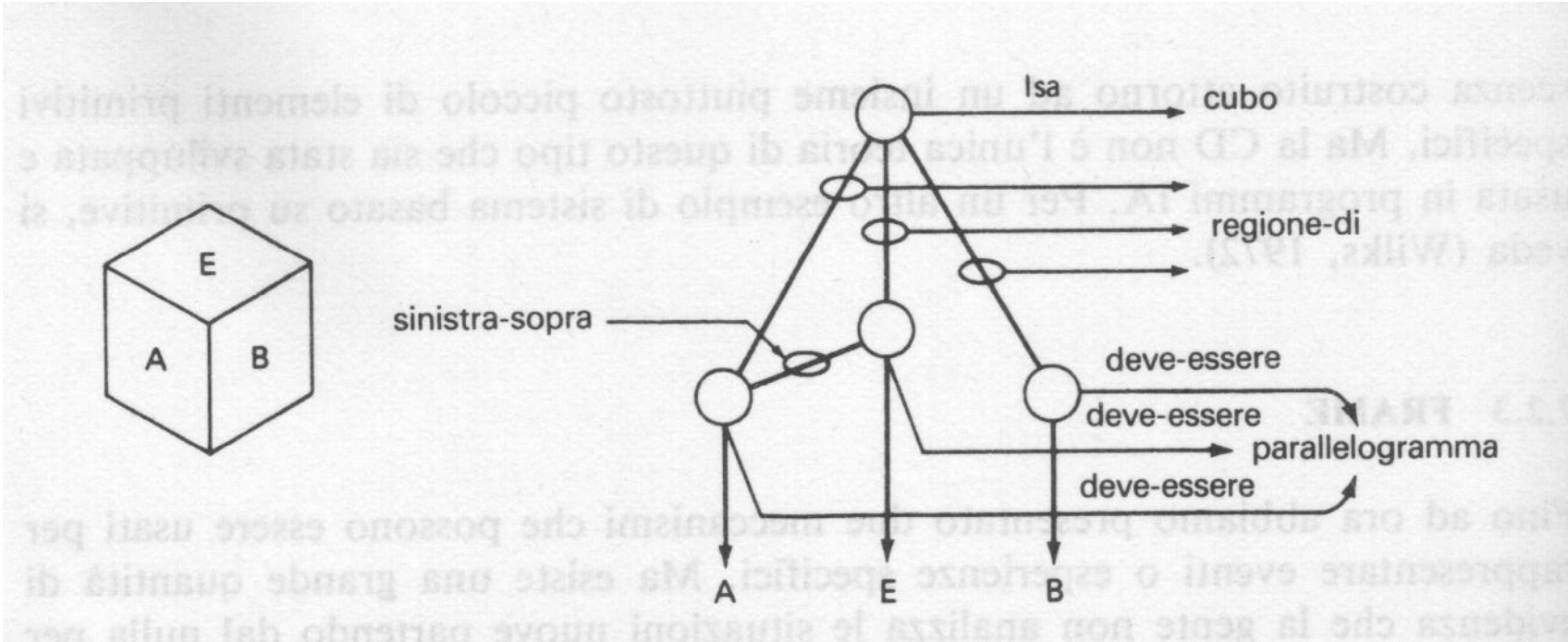
- È costituito da una collezione di caselle (slot) che descrivono aspetti degli oggetti.

- Gli slot possono essere riempiti con altri frame che descrivono altri oggetti, ovvero con valori di default.

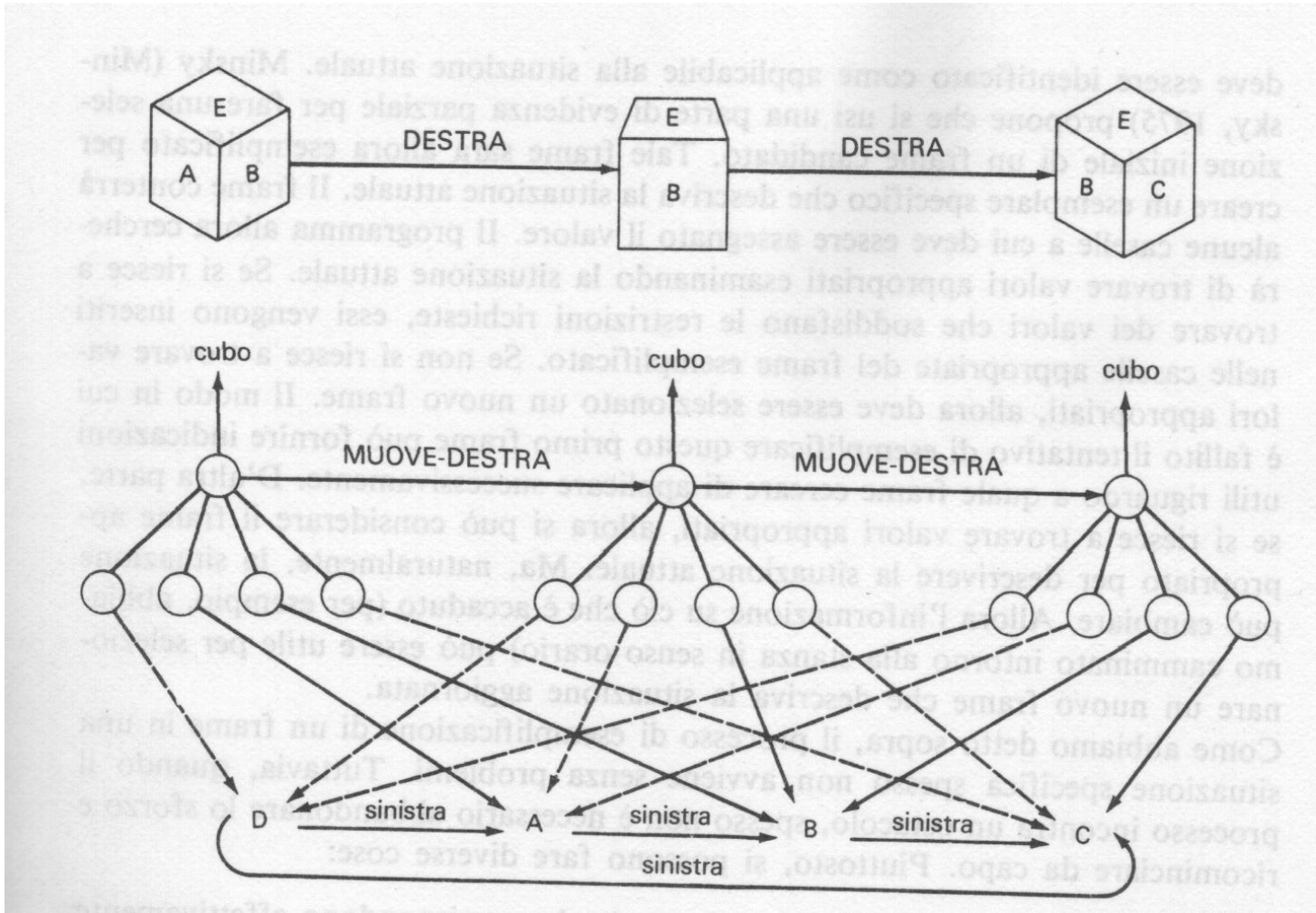
- Ad uno slot può essere associato un insieme di condizioni che devono essere soddisfatte dall'elemento che le riempia

- Ad uno slot può essere associata anche informazione procedurale
  - Procedura if-added: descrivono che cosa si deve fare ogni volta che uno slot è riempito
  - Procedure if-needed o to-establish: descrivono come si può calcolare, se richiesto, l'elemento da usare per il riempimento. Si parla di associazione procedurale (*procedural attachment*).
- Una situazione complessa è rappresentata da frame correlati, cioè un sistema di frame.

Esempio: rappresentazione di un (singolo) cubo (da un particolare punto di vista):



Per rappresentare l'oggetto cubo occorrono più viste: occorre un sistema di frame:



- I frame sono strutture complesse che ricalcano la complessità delle strutture concettuali: quando si richiama un concetto, e quindi un frame, si attivano anche i concetti connessi (i frame contenuti nel frame centrale)

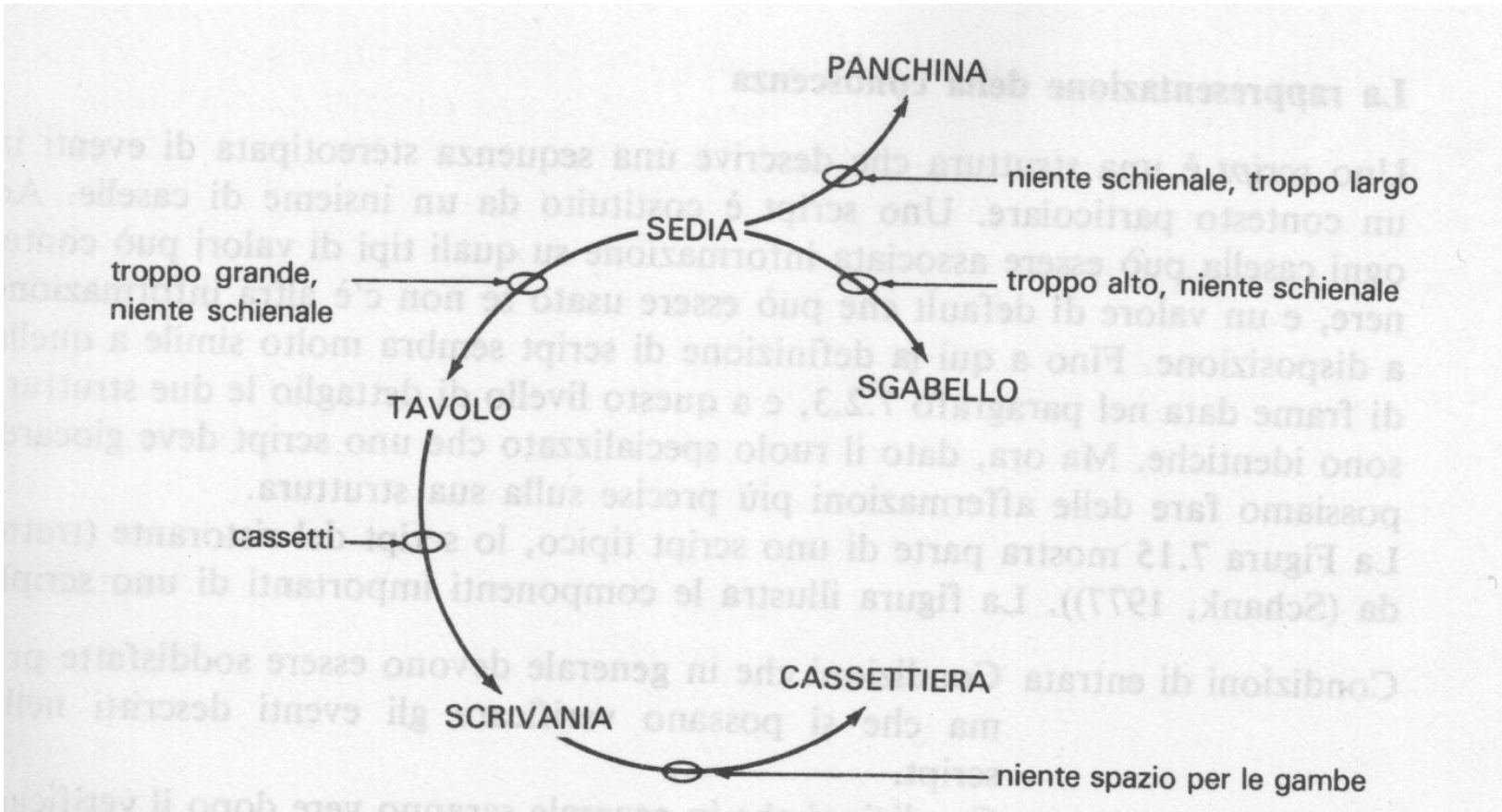
- La struttura a frame può essere usata per il riconoscimento (di oggetti o frasi): i frame contengono attributi degli oggetti che devono essere veri e che saranno usati per riempire caselle individuali. Le procedure associate agli slot “guidano” il processo di riconoscimento.

- I frame descrivono esemplari tipici dei concetti che rappresentano. Ciò risulta utile quando non sono disponibili, o non si sa come accedere, a tutti gli elementi. Alcuni di questi possono essere “supposti” perché contenuti nel frame. Caso tipico: riconoscimento di oggetti con parti nascoste (visione, parlato, ecc.)

Come si usano i frame?

- Si utilizza una parte di evidenza parziale per selezionare un frame
- Si cerca di riempire gli slot vuoti
- Se la fase precedente fallisce, le “modalità del fallimento” possono dare informazioni su quale altro frame selezionare (si utilizzano cioè le “motivazioni” del fallimento).

- Si utilizzano i legami specifici tra frame per individuare le direzioni da esplorare, come, ad esempio, una rete di somiglianze:



- Si può risalire la struttura gerarchica in cui eventualmente sono inseriti i frame (esempio cane → mammifero → animale), finché si trova il fame che unifica (deve essere prevista la possibilità di arricchire la struttura con nuovi frame, quando si rilevano elementi non previsti nei frame precedenti: esempio presenza baffi aggiungere frame gatto!)

Script

- È una struttura che descrive una sequenza stereotipata di eventi
- È anch'essa costituita da slot, a cui è associata l'informazione su quali tipi di valori può contenere o i valori di default
- Si differenzia dai frame in quanto ha un ruolo specializzato

Componenti rilevanti in uno script:

Condizioni di entrata: Condizioni che in generale devono essere soddisfatte prima che si possano verificare gli eventi descritti nello script.

Risultato: Condizioni che in generale saranno vere dopo il verificarsi degli eventi descritti nello script.

Props Caselle che rappresentano oggetti che sono implicati negli eventi descritti nello script. La presenza di questi oggetti può essere inferita anche se non sono menzionati esplicitamente.

Ruoli Caselle che rappresentano persone che sono coinvolte negli eventi descritti nello script. Anche la presenza di queste persone può essere inferita anche se non sono menzionate esplicitamente. Se sono menzionati individui specifici, possono essere inseriti nelle caselle appropriata.

Traccia La variazione specifica rispetto ad uno schema più generale che è rappresentato da questo script particolare. Tracce diverse dello stesso script condivideranno molte componenti ma non tutte.

Scene Le effettive sequenze di eventi che si verificano. Gli eventi sono rappresentati nel formalismo della dipendenza concettuale.

Esempio:

Lo script “Ristorante”

Script: RISTORANTE
Traccia: Bar
Props: Tavoli
Menù
F = Cibo
Controllo
Soldi

Ruoli: S = Cliente
W = Cameriere
C = Cuoco
M = Cassiere
O = Proprietario

Condizioni di entrata:
S è affamato
S ha denaro

Risultati:
S ha meno denaro
O ha più denaro
S non è affamato
S è soddisfatto
(opzionale)

Scena 1: Ingresso

S PTRANS nel ristorante
S ATTEND occhio ai tavoli
S MBUILD dove sedere
S PTRANS S al tavolo
S MOVE S in posizione seduta

Scena 2: Ordinazione

(Menù sul tavolo)
(W porta il menù)
S PTRANS menù a S

(S chiede il menù)
S MTRANS segnala a W
W PTRANS W al tavolo
S MTRANS 'bisogno menù' a W
W PTRANS W al menù

W PTRANS W al tavolo
W ATRANS menù a S

S MTRANS menù a CP(S)
*S MBUID scelta di F
S MTRANS segnale a W
W PTRANS W al tavolo
S MTRANS 'voglio F' a W

W PTRANS a C
W MTRANS (ATRANS F) a C

C MTRANS 'no F' a W
W PTRANS W a S
W MTRANS 'no F' a S
(torna indietro a*) o
(va alla Scena 4 per l'azione "senza pagare")

C DO (prepara F script)
va alla Scena 3

Scena 3: Mangiare

C ATRANS F a W
W ATRANS F a S
S INGEST F

(Opzione: Ritorna alla Scena 2 per ordinare ancora; altrimenti va alla Scena 4)

Scena 4: Uscita

S MTRANS a W
(W ATRANS assegno a S)

W MOVE (fa l'assegno)
W PTRANS W a S
S ATRANS assegno a S
S ATRANS mancia a W
S PTRANS S a M
S ATRANS denaro a M
S PTRANS S fuori dal ristorante

(Azione "senza pagare")

Gli script descrivono dunque catene causali di eventi. La loro forza risiede nella capacità di predire eventi non osservati esplicitamente.

Esempio

La seguente storia:

John ieri sera uscì per andare in un ristorante. Ordinò una bistecca. Quando pagò, notò che stava per finire i soldi. Poiché aveva iniziato a piovere corse a casa.

attiva e soddisfa lo script del ristorante.

Quindi alla domanda :

John cenò ieri sera?

che non unifica con nessuna parte della storia, si può rispondere positivamente, in quanto è un evento previsto dallo script.

Lo script permette di eseguire correlazioni causali, e quindi di costruire interpretazioni coerenti a partire da collezioni di osservazioni, come nel seguente esempio.

Sia fornita la seguente storia:

John uscì a pranzo. Si sedette a tavola e chiamò la cameriera. La cameriera gli portò un menù ed egli ordinò un hamburger.

Ora consideriamo la domanda

Perché la cameriera portò a John il menù?

Lo script fornisce due possibili risposte a questa domanda:

- Perché John glielo ha chiesto (Questa risposta si ottiene andando indietro nella catena causale per individuare che cosa ha causato l'azione della cameriera).

- Perché John potesse decidere che cosa voleva mangiare (Questa risposta si ottiene andando in avanti nella catena causale per individuare quale evento è permesso dall'azione della cameriera).

Come i frame, contiene meccanismi per la focalizzazione dell'attenzione (in questo caso, su eventi inusuali).

Esempio:

John andò al ristorante. Gli venne assegnato un tavolo. Ordinò una grossa bistecca. Rimase seduto ad aspettare per molto tempo. Si arrabbiò e uscì.

"Si arrabbiò" non è previsto dallo script Ristorante, interrompe l'uso di quello script e sposta l'attenzione su un altro fatto.