Laurea Specialistica in Informatica
a.a. 2005-2006

Interazione Uomo-Macchina II:

Interfacce Intelligenti

Fiorella de Rosis

Introduzione

Prima parte: *Formalizzazione e Ragionamento*

1.1. Ragionamento logico:
- Formalizzazione
- Risoluzione

1.2. Ragionamento incerto
- Reti Causali Probabilistiche
- Reti dinamiche
- Apprendimento di Reti

Seconda parte: *Modelli di Utente*

2.1. Modelli logici

2.2. Modelli con incertezza

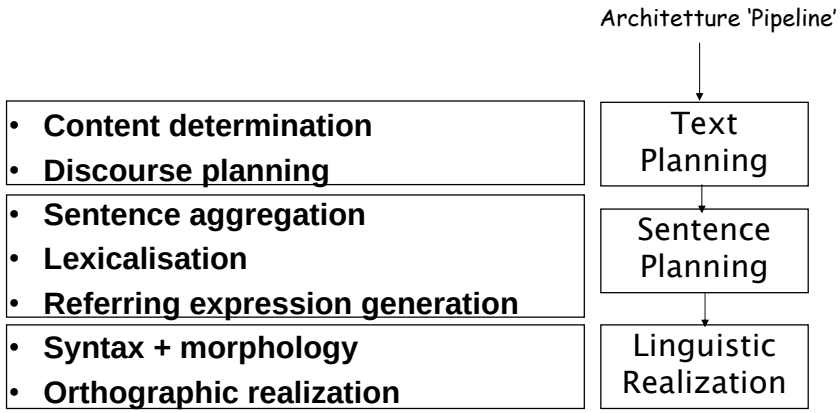
Terza parte: *Interazione in linguaggio naturale*

3.1. Generazione di messaggi
- Introduzione
- Teorie
- ***Metodi***

3.2. Comprensione di messaggi

Quarta parte: *Simulazione di dialoghi*

Task nell' NLG



Nelle architetture pipeline,
le diverse fasi della generazione di un messaggio
si susseguono in ordine prestabilito

La "nostra" architettura di riferimento

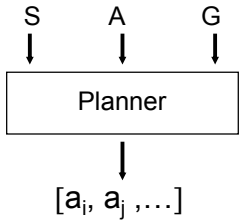
Due Task Principali

- ***definizione del 'piano del discorso':***
  cosa dire e in che ordine dirlo;
- ***generazione 'superficiale'*:**
  come dirlo.

… vediamo cosa s'intende per 'pianificazione'…

1. Definizione di Pianificazione:

E' il processo di definizione della lista ordinata di azioni da compiere per raggiungere un determinato *stato-obiettivo G*, a partire da una determinata *situazione iniziale* S e dato un insieme A= $[a_1, a_2, \ldots, a_n]$ di *azioni disponibili.*



Un algoritmo di pianificazione richiede una descrizione simbolica coerente di S, di G e degli elementi di A.

Introduzione alla Pianificazione: Metodo di Green

Una procedura di pianificazione basata sul principio di risoluzione.

Argomenti:

- un termine che denota lo *stato iniziale*
- una descrizione dello *stato iniziale*
- una descrizione dello *stato-goal*
- una descrizione (generale) degli *operatori di piano* disponibili,

il Metodo di Green applica il principio di risoluzione, con metodo *fill-in the blank* e strategia ordinata, per derivare il *piano* che conduce dallo stato iniziale allo stato finale.

Il piano è una sequenza di operatori, instanziati.

Premessa:

Rappresentazione di stati, azioni e loro effetti

Denotiamo con S_i uno *stato*.

Introduciamo un operatore T applicato a una formula e un termine $T(\varphi, S_i)$ per indicare che
'la proprietà φ è vera nello stato S_i'.
Queste relazioni potranno essere utilizzate, in particolare, per *descrivere lo 'stato iniziale'* e lo *'stato goal'.*

Denotiamo con A una *azione*.
Introduciamo la funzione Do
che mette in relazione: coppie (azione, stato) a stati:

$$Do(A, S_h) \rightarrow S_k$$

Introduciamo il predicato Goal per denotare uno stato come stato-goal: Goal(S_k) indica che S_k è lo stato-goal.

Premessa:

Descrizione delle azioni (operatori)

Descriviamo le azioni in termini di precondizioni ed effetti:

precondizione = formula che descrive lo stato S_h *prima* dell'applicazione dell'operatore

effetto = formula che descrive lo stato s_k *dopo* l'applicazione dell'operatore, con S_k descritto mediante la funzione Do:

$Do(A, s_h) = S_k$

$$(T(\varphi_1,S_h) \wedge \ldots \wedge T(\varphi_n,S_h)) \rightarrow (T(\varphi_{n+1},Do(A,S_h)) \wedge T(\varphi_{n+m},Do(A, S_h)))$$

"*se nello stato* S_h *sono vere le proprietà* $\varphi_1, \ldots \varphi_n$
e si applica l'operatore A,
si produce uno stato S_k *in cui valgono le proprietà*
$\varphi_{n+1}, \ldots, \varphi_{n+m}$

('situation calculus')

Premessa:
Descrizione dello stato-goal

Descriviamo lo stato-goal con le proprietà che sono vere
In quello stato:

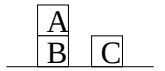
$(T(\varphi_1,t) \wedge \ldots \wedge T(\varphi_n,t)) \Leftrightarrow Goal(T,t)$

" *nello stato goal (che denotiamo con t) sono vere le proprietà* φ_1 and … φ_n

Un esempio nel mondo dei blocchi:
Rappresentazione di stati, operatori e loro effetti

Riprendiamo il formalismo che abbiamo introdotto nell'Unità 2 per descrivere il mondo dei blocchi

S_i : stato iniziale
$T(On(A,B),S_i) \wedge T(Clear(A), S_i)$



S_i

Descriviamo (in modo generale) l'operatore di Unstack:
Unstack(x,y) = 'unstack di x da y'
$(T(On(x,y), S_i) \wedge T(Clear(x), S_i)) \rightarrow$
$(T(Table(x),Do(Unstack(x,y),S_i)) \wedge T(Clear(y),Do(Unstack(x,y),S_i))$

l'applicazione della unstack ad A e B' Unstack(A,B) può essere descritta come:
$(T(On(A,B), S_i) \wedge T(Clear(A), S_i)) \rightarrow$
$T(Table(A),Do(Unstack(A,B),S_i)) \wedge T(Clear(B),Do(Unstack(A,B),S_i)$

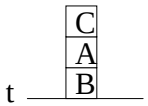
Notare: la descrizione degli stati non è necessariamente completa!!!

Un esempio nel mondo dei blocchi:
Rappresentazione di stati, azioni e loro effetti

Descriviamo (analogamente) l'operatore di Stack:
Stack(x,y) = 'stack di x su y'
$(T(Table(x), S_i) \wedge T(Clear(x), S_i) \wedge T(Clear(y), S_i)) \rightarrow$
$T(On(x,y),Do(Stack(x,y),S_i)$

"l'operatore di stack di x su y può essere applicato se x è sul tavolo e entrambi i blocchi sono clear'; produce, come effetto, che x sta sopra ad y".

Descriviamo lo stato goal:
$(T(On(C,A),t) \wedge T(On(A,B),t)) \Leftrightarrow Goal(t)$
"*nello stato goal t, il blocco C è sul blocco A e il blocco A è sul blocco B"*

t 

Pianificazione con il Metodo di Green

Si rappresentano in un dataset, in forma di clausole:
stato iniziale e operatori disponibili.

Si pone il quesito: qual'è la concatenazione di operatori che permette di ottenere lo stato goal?

Per rispondere a questo quesito:
si aggiunge al dataset il goal negato
si applica il principio di risoluzione nella modalità 'fill-in the blank'.

I termini della (o delle) clausole 'Ans' derivate indicano il (o i) piani individuati.

Pianificazione con il Metodo di Green

un esempio semplice

$T(On(A,B), S_1)$
$T(Clear(A), S_1)$
$(T(Table(x),s) \wedge T(Clear(x),s) \wedge T(Clear(y), s)) \rightarrow T(On(x,y), Do(Stack(x,y),s))$
$(T(On(x,y),s) \wedge T(Clear(x), s)) \rightarrow (T(Table(x), Do(Unstack(x,y),s))$
$(T(On(x,y),s) \wedge T(Clear(x), s)) \rightarrow (T(Clear(y), Do(Unstack(x,y),s))$
$T(Table(A), t) \Leftrightarrow Goal(t)$

Convertiamo in clausole, *neghiamo il goal e aggiungiamo il letterale Ans*:
1 $\{T(On(A,B), S_1)\}$
2 $\{T(Clear(A), S_1)\}$
3 $\{\neg T(Table(x),s), \neg T(Clear(x),s), \neg T(Clear(y), s)), T(On(x,y), Do(Stack(x,y),s))\}$
4 $\{ \neg T(On(x,y),s), \neg T(Clear(x), s)), T(Table(x), Do(Unstack(x,y),s))\}$
5 $\{ \neg T(On(x,y),s), \neg T(Clear(x), s)), T(Clear(y), Do(Unstack(x,y),s))\}$
6 $\{\neg T(Table(A), Do(a, S_1)), Ans(a))\}$
Risolvo con strategia 'ordinata':
(1,4), con x/A, y/B e s/ S_1 e ottengo la 7
(2,7), con x/A e ottengo la 8
(5,8), con x/A e ottengo: $\{Ans(Unstack(A,B))\}$

Esercizio

Riprendi l'esempio precedente, ma aggiungi le seguenti condizioni

- sullo stato iniziale:
  A e B sono rossi, mentre C è verde
- sullo stato-goal:
  mettere tutti i blocchi rossi sul tavolo

Esercizio

Riprendi l'esempio precedente,
cambiando gli operatori nel modo seguente:

R-Unstack(x,y):	unstack di un blocco rosso da un blocco verde
G-Unstack(x,y):	unstack di un blocco verde da un blocco rosso
R-Stack(x,y):	stack di un blocco rosso su un blocco verde
G-Stack(x,y):	stack di un blocco verde su un blocco rosso

Direzione della ricerca della soluzione

Nell'esempio precedente, la risoluzione ordinata era effettuata in modo 'forward':
1 $\{T(On(A,B), S_1)\}$
2 $\{T(Clear(A), S_1)\}$
3 $\{\neg T(Table(x),s), \neg T(Clear(x),s), \neg T(Clear(y), s)), T(On(x,y), Do(Stack(x,y),s)\}$
4 $\{\neg T(On(x,y),s), \neg T(Clear(x), s)), T(Table(x), Do(Unstack(x,y),s)\}$
5 $\{\neg T(Table(A), Do(a, S_1)), Ans(a))\}$

Ma si può simulare una ricerca della soluzione backward,
a partire dalla descrizione del goal:

1 $\{T(On(A,B), S_1)\}$
2 $\{T(Clear(A), S_1)\}$
3 $\{T(On(x,y),Do(Stack(x,y),s)),\neg T(Table(x),s),\neg T(Clear(x),s),\neg T(Clear(y), s))\}$
4 $\{T(Table(x), Do(Unstack(x,y),s)), \neg T(On(x,y),s), \neg T(Clear(x), s))\}$
5 $\{\neg T(Table(A), Do(a, S_1)), Ans(a))\}$
In questo caso:
(4,5) con x/A produce:
$\{\neg T(On(A,y),S_1),\neg T(Clear(A),S_1),Ans(Unstack(A,y),S_1\}$, 6
(1,6) con y/B produce la 7
(2,7) produce $\{Ans(Unstack(A,B)\}$

Esercizio

Stesso stato iniziale dell'esercizio precedente.
Ma A e B sono rossi mentre C è verde.
Mettere tutti i blocchi rossi sul tavolo.

Pianificazione con il Metodo di Green:
un esempio più complesso

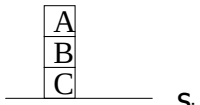
Nel caso precedente, bastava *una sola azione* per passare dallo stato iniziale allo stato-goal (esempio poco realistico).
Le cose si complicano, se occorre una *concatenazione di azioni* per ottenere il risultato.

Definisco: concatenazione di azioni una lista ordinata
$[a_1, a_2, \ldots, a_n]$ tale che:

$Do([a_1, a_2, \ldots, a_n],s) = Do(a_n,Do(a_{n-1},\ldots Do(a_1,s))\ldots)$.

Consideriamo questo esempio :

S_i : stato iniziale
$T(On(A,B),s_i) \wedge T(Clear(A), s_i) \wedge T(Table(C), s_i) \wedge T(On(B,C), s_i)$



S_i

t: stato-goal
T(Table(B), t)

t

la concatenazione di azioni necessaria è, in questo caso:
[Unstack(A,B), Unstack(B,C)]

… sviluppiamolo …

1 {T(On(A,B), S_1)}
2 {T(Clear(A), S_1)}
3 {T(On(B,C), S_1)}
4 {T(Table(C), S_1)}

5 {¬ T(Table(B), Do(a, S_1)), Ans(a)}

6 {¬T(On(x,y),s), ¬T(Clear(x), s)), T(Table(x), Do(Unstack(x,y),s))}
7 {¬T(On(x,y),s), ¬T(Clear(x), s)), T(Clear(y), Do(Unstack(x,y),s))}
==========
8 {¬T(Clear(A), s)), T(Table(A), Do(Unstack(A,B), S_1))} (1,6), x/A,y/B
9 {T(Table(A), Do(Unstack(A,B), S_1))} (2,8)
10 {¬T(Clear(A), s)), T(Clear(B), Do(Unstack(A,B), S_1))} (1,7)
11 {T(Clear(B), Do(Unstack(A,B), S_1))} (2,10)

Riapplico lo stesso procedimento con On(B,C) :

12 {¬T(Clear(B), s)), T(Table(B), Do(Unstack(B,C), S_1))}

… ma per risolvere la 11 con la12 ho bisogno di un metodo per rappresentare la concatenazione delle azioni…

Il 'Frame problem'

Descrivere gli aspetti di uno stato
che non vengono cambiati dall'applicazione di un operatore.

Per l'operatore di Unstack:

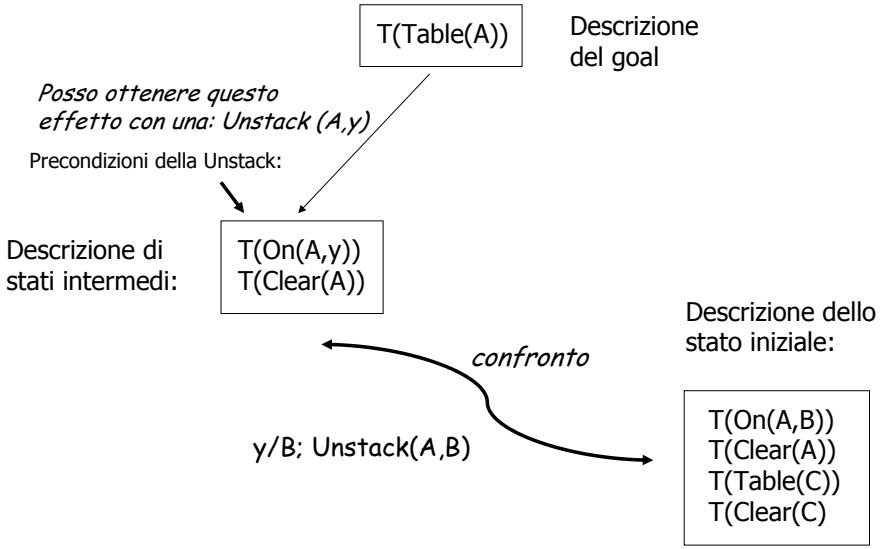
$T(Clear(u),s) \Rightarrow T(Clear(u),Do(U(x,y),s))$
$T(Table(u),s) \Rightarrow T(Table(u),Do(U(x,y),s))$
$T(On(u,v),s) \wedge u \neq x \Rightarrow T(On(u,v),Do(U(x,y),s))$

Per l'operatore di Stack:

$T(Clear(u),s) \wedge u \neq y \Rightarrow T(Clear(u),Do(S(x,y),s))$
$T(Table(u),s) \wedge u \neq x \Rightarrow T(Table(u),Do(U(x,y),s))$
$T(On(u,v),s) \Rightarrow T(On(u,v),Do(U(x,y),s))$

… ma non è chiaro a che serve…

Rappresentiamo graficamente la risoluzione backward



2. Pianificazione per 'Regressione dal Goal'

Una procedura di pianificazione che utilizza

- un diverso formalismo per rappresentare gli operatori e gli stati
- un diverso metodo di ricerca della soluzione
  (backward, e cioè a partire dal goal, fino a raggiungere lo stato iniziale).

Operatore A:

Pre(A):	lista di condizioni che devono essere verificate perché l'operatore sia applicabile;
Add(A):	lista di proprietà che diventano vere dopo l'applicazione dell'operatore
Del(A):	lista di proprietà che diventano false dopo l'applicazione dell'operatore

Un esempio nel mondo dei blocchi:
il formalismo

Unstack(x,y)

Pre(Unstack(x,y)):	$(On(x,y) \wedge Clear(x))$
Add(Unstack(x,y)):	$(Table(x) \wedge Clear(y))$
Del(Unstack(x,y)):	On(x,y)

Stack(x,y)

Pre(Stack(x,y)):	$(Table(x) \wedge Clear(x) \wedge Clear(y))$
Add(Stack(x,y)):	On(x,y)
Del(Stack(x,y)):	(Table(x),Clear(y))

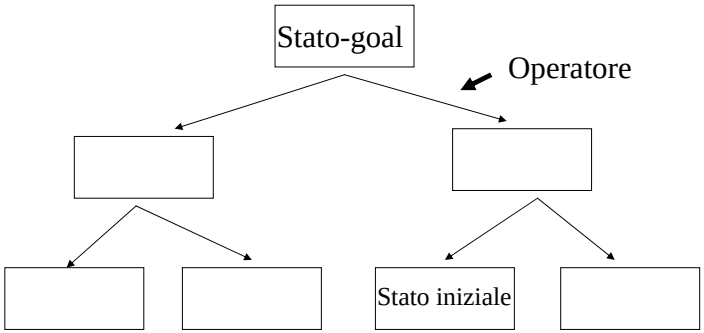
Stato S_1

$On(A,B) \wedge Clear(A) \wedge Table(C) \wedge Clear(C)$

Stato-goal:

On(C,A)

Metodo di ricerca della soluzione



Metodo di ricerca della soluzione

A partire dallo stato goal,
si applica ricorsivamente al generico stato q una procedura che:

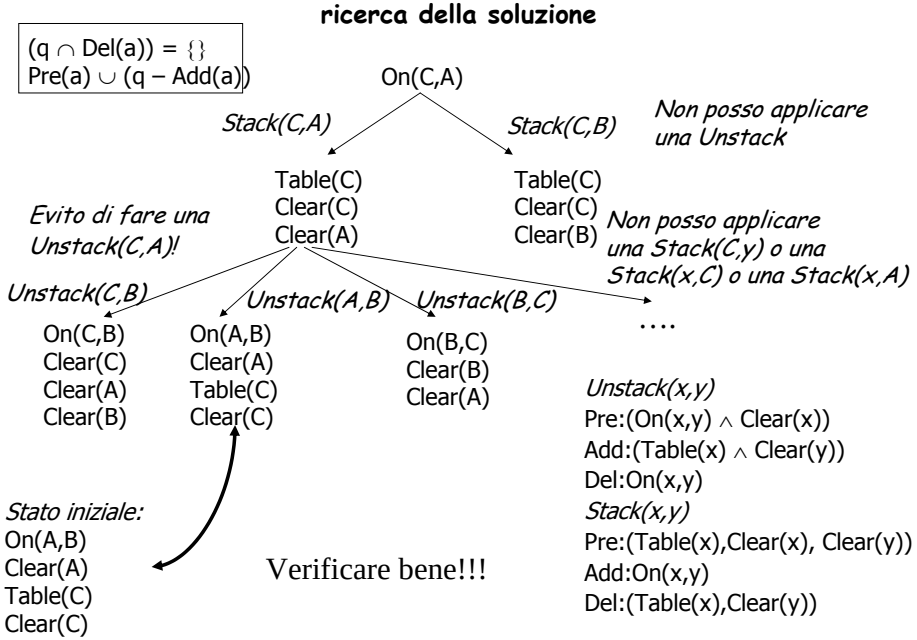
- cerca un operatore a applicabile
  (come l'operatore la cui parte Del ha una intersezione vuota con la descrizione di q),
- determina il nuovo stato Reg(q,a):
  (come lo stato descritto dall'unione di Pre(a) e q-Add(a))

$$(q \cap Del(a)) = \{\} \Rightarrow \; Reg(q,a) = Pre(a) \cup (q - Add(a))$$

Criterio:
- applicare operatori che evitino di cancellare *proprietà da raggiungere*,
- descrivere il nuovo stato-goal mediante l'insieme delle *precondizioni dell'operatore* e delle *proprietà non ancora raggiunte*.

L'esempio nel mondo dei blocchi:

ricerca della soluzione



Pianificazione di discorsi

Un atto comunicativo può essere visto come un'azione che uno Speaker (S) compie nei confronti di un ascoltatore (H) con l'*obiettivo di cambiare il suo stato mentale.*

Esempio:
- con una *'inform'* su un determinato fatto, S si pone l'obiettivo di far sì che H creda in quel fatto,
- con una *'answer'* su un determinato fatto, S si pone lo stesso obiettivo,
- con una *'suggest'* su una azione, si pone l'obiettivo che H compia quella azione, ecc.

Ma dietro ogni azione c'è anche una *ipotesi di S sullo stato mentale di H*:
- dietro una *'inform'*, c'è l'ipotesi che H non creda già in quel fatto,
- dietro una *'answer'*, c'è l'ipotesi che H desideri conoscere il valore di verità del fatto e che non lo conosca già (ad es perché H gliel'ha chiesto),
- dietro una *'suggest'* c'è l'ipotesi che H non abbia già l'intenzione di compiere quell'azione, ecc.

Pianificare un discorso, quindi, non è molto diverso dal pianificare una qualsiasi attività complessa.

Un modo di vedere il problema della pianificazione di discorsi

Descrizione degli stati:

Combinazione logica di elementi dello stato mentale dell'utente (come nell'Unità 4):

predicati: KnowAbout, WantToKnow, Prefer, Like, IsInterestedIn, KnowHow, CanDo, oppure belief

Descrizione degli operatori:

Atti comunicativi:

Inform(S,U,f): informare su un fatto f

Persuade(S,U,a): persuadere a compiere una azione a

Instruct(S,U,a): istruire su come compiere una azione a

…

descritti come operatori (Pre & Add & Del)

Il Piano del Discorso consiste in una semplice concatenazione di atti comunicativi che permette di realizzare lo stato-goal a partire dallo stato-iniziale

Pianificazione di discorsi:

un esempio

Descrizione dello stato-goal:

l'utente deve avere l'intenzione di compiere l'azione A e deve sapere come farlo

WantToDo(U,A)
KnowHow(U,A)

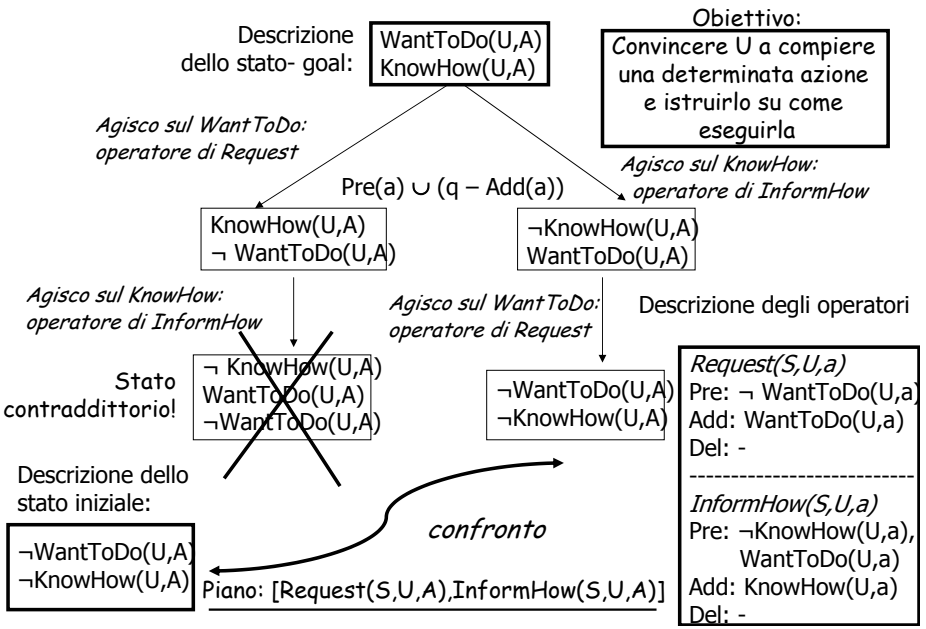
Descrizione dello stato iniziale:

¬WantToDo(U,A)
¬KnowHow(U,A)

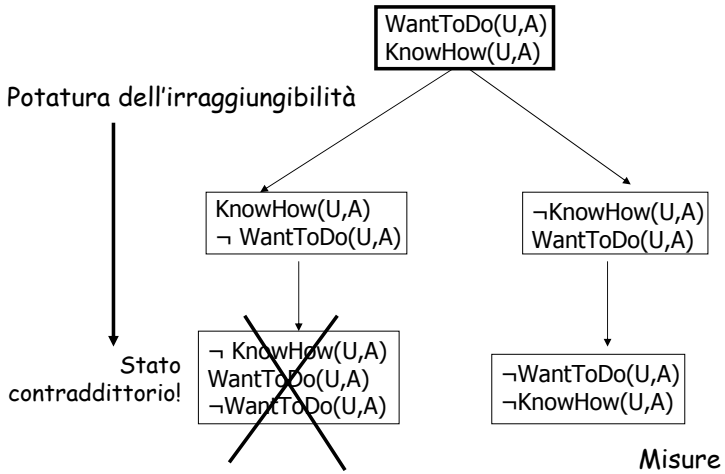
Descrizione degli operatori:

Request(S,U,a)
Pre: ¬ WantToDo(U,a)
Add: WantToDo(U,a)
Del: -
==========
InformHow(S,U,a)
Pre: ¬KnowHow(U,a),
WantToDo(U,a)
Add: KnowHow(U,A)
Del: -

Pianificazione di discorsi per regressione dal goal

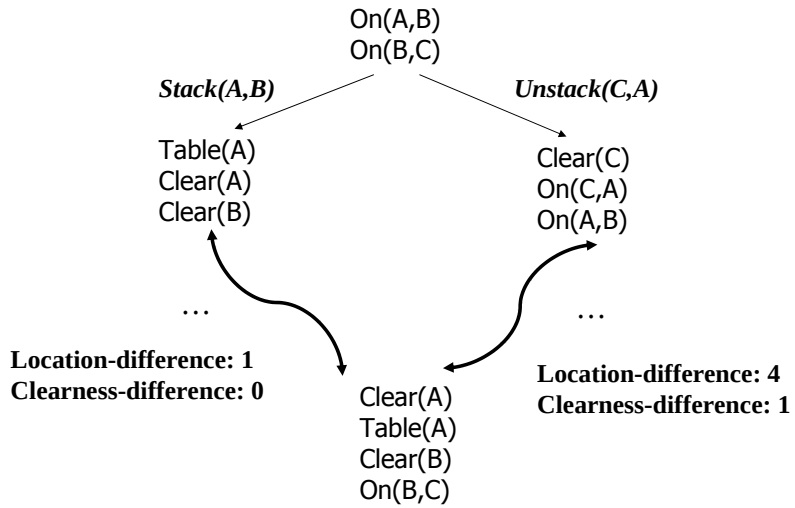


Metodi per la riduzione dello spazio di ricerca:



Misure di 'somiglianza' fra gli stati e ordinamento degli stati da considerare ad ogni passo, in funzione del grado di somiglianza rispetto allo stato iniziale

Esempio di misura di somiglianza nel mondo dei blocchi

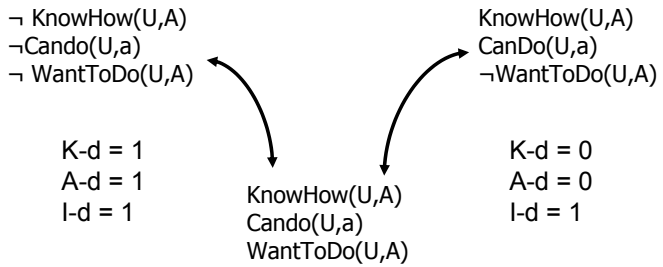


Misure di somiglianza nel dominio del discorso

Consistono nel 'misurare' la distanza fra stati in termini di differenza fra gli elementi dello stato mentale dell'utente:

Knowledge-difference
Intention-difference
Ability-difference
…ecc

Esempio:



Quali problemi nel metodo di regressione dal goal?

1. Lo spazio di ricerca può essere molto complesso,
   e quindi la ricerca della soluzione può richiedere uno spazio e un tempo inaccettabili, se il problema non è banale
   (goal-state 'ricco' e molti atti comunicativi disponibili).
2. Ci possono essere molti modi per ottenere uno stesso effetto:
   ad esempio, una Request si può formulare in termini di Order, di Suggest, di Praise,…

Soluzione:

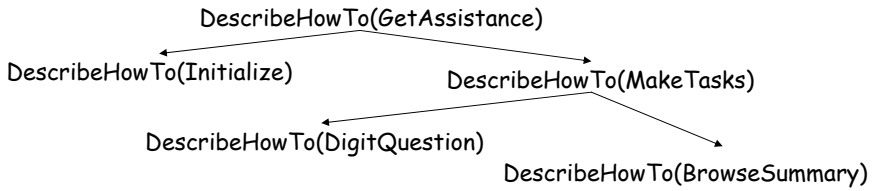
applicare un metodo di pianificazione diverso,
basato su una *ricerca di soluzioni 'per livelli di astrazione'*.

Cioè, decomporre il problema in sottoproblemi
(il goal in sub-goal)
fino ad individuare problemi semplici,
affrontabili con un solo operatore

Un esempio: Generazione di messaggi di help in Word

Ottenere assistenza dal menu ?

Scegliere Guida in linea Microsoft Word dal menu ?.
Se è abilitato, verrà visualizzato l'Assistente.
Se l'Assistente è disabilitato, verrà visualizzata la finestra della Guida.
Per digitare una domanda nella finestra della Guida,
fare clic sulla scheda Ricerca libera.
Per sfogliare il sommario della Guida,
fare clic sulla scheda Sommario. …



DescribeHowTo(task): … *browse summary*
InformAbout(S,U,Purpose(task)) … *sfogliare il sommario della Guida*
Suggest(S,U,Action(task)) …*fare click sulla scheda Sommario*

Si formulano gli stati in modo 'astratto' anziché per descrizione dello stato mentale dell'utente:

DescribeHowTo(GetAssistance) si decompone in:
DescribeHowTo(Initialize) e
DescribeHowTo(MakeTasks)

DescribeHowTo(MakeTasks) si decompone in:
DescribeHowTo(DigitQuestion)
DescribeHowTo(BrowseSummary)

fino a decomporre in 'azioni elementari' che corrispondono a singoli 'atti comunicativi'. Questi vengono descritti in termini di precondizioni ed effetti:

InformAbout(S,U,x), con x=formula
condition: ¬KnowAbout (U,x)
effect: KnowAbout (U,x)

Suggest(S,U,a), con a=termine
condition: ¬WantToDo (U,x)
effect: WantToDo (U,x)

Relazione con la Teoria

Cosa ritroviamo in questo metodo di pianificazione, delle teorie che abbiamo visto nell'Unità precedente?

- La *struttura linguistica* di un piano è data dall'insieme delle sue foglie (le frasi che lo costituiscono);
- La sua *struttura intenzionale* è rispecchiata dalla struttura di goal e sub-goal, che indicano con chiarezza le relazioni di precedenza fra i discourse segment DS;
- La sua *struttura attenzionale* (focus dei diversi discourse segment) è rispecchiata dai termini dei goal/subgoal o degli atti comunicativi elementari. Nell'esempio precedente: GetAssistance, MakeTask, DigitQuestion, ecc;
- Le diverse parti del piano sono connesse fra loro da *relazioni retoriche;*
- *I singoli* speech act sono descritti in termini di precondizioni ed effetti.

Operatori per la Pianificazione del Discorso

Sono basati su un'estensione dell'idea di rappresentazione di azioni come frame.

Name:	*un identificatore dell'operatore*
Header:	*la descrizione del suo obiettivo*
Constraints:	*vincoli (su S e su H) che devono essere soddisfatti perché l'operatore sia applicabile*
Effects:	*effetti dell'operatore sulla mente di H*
Decomposition:	*la descrizione dei sotto-obiettivi in cui l'obiettivo descritto nello Header può essere decomposto*

Il contenuto dei vari slot è una wff.

Generazione di Testi: Operatori di Moore e Paris

Ogni operatore contiene i seguenti attributi:

- **Effetto:** obiettivo dell'azione descritta dall'operatore
- **Una lista di vincoli (precondizioni):** condizioni che devono essere vere per poter applicare l'operatore e „garantire" che produca l'effetto descritto. Queste precondizioni possono riferirsi a fatti nella base di conoscenza sul dominio, nello user model, nella storia del dialogo oppure sullo stato del processo di pianificazione.
- **Un nucleo:** rappresenta il sottogoal più „importante" in cui si decompone il goal. E' obbligatorio
- **Uno o più satelliti:** rappresentano sottogoal aggiuntivi che contribuiscono a raggiungere l'effetto dell'operatore. Possono essere opzionali.

Operatori di Moore e Paris

Esempio: operatore di piano per la MOTIVATION

- **EFFECT: (MOTIVATION ?act ?goal)**
- **CONSTRAINTS: (AND (STEP ?act ?goal)**
- **(GOAL ?hearer ?goal))**
- **NUCLEUS: (BEL ?hearer (STEP ?act ?goal))**
- **SATELLITES: NIL**

Il pianificatore:

- utilizza unm odello degli stati mentali e dei goal comunicativi:
  - **(KNOW ?agent (ref ?description))**
  - **(BEL ?agent (?predicate ?e1 ?e2))**
- funziona per espansione gerarchica top-down.

Generazione di Messaggi Multimodali

(M.Maybury, MITRE Corporation)

Gli atti comunicativi linguistici vengono estesi
con atti 'non verbali':
ad esempio, atti 'fisici' (del corpo)
e 'grafici' (disegni o immagini)

PHYSICAL ACTS	LINGUISTIC ACTS	GRAPHICAL ACTS
DEICTIC ACT		
Point, tap, circle	**Inform,…**	**Highlight, blink, circle**
Indicate direction	**Suggest,…**	**Indicate direction**
ATTENTIONAL ACT	**Ask,…**	**DISPLAY CONTROL ACT**
Snap/clap fingers, clap hands		**Display-region** **Zoom (in, out)**
BODY LANGUAGE ACT		**DEPICT ACT**
Facial expressions		**Depict image**
gestures		**Draw (line, arc, circle,…)**
Sign language		**Animate-action**

Operatori Multimodali di Maybury (1)

==========

NAME: *Identify-location-linguistically*
HEADER: Identify(S,H, e)
CONSTRAINTS: Entity(e)
PRECONDITIONS: Goal S, KnowAbout(H, Location°(e))
EFFECTS: KnowAbout(H, Location°(e))
DECOMPOSITION: Inform(S,H, Location°(e)))

==========

NAME: *Identify-location-linguistically&visually*
HEADER: Identify(S,H, e)
CONSTRAINTS: CartographicEntity(e)
PRECONDITIONS: Visible(e) $\wedge$ Goal S, KnowAbout(H, Location°(e))
EFFECTS: KnowAbout(H, Location°(e))
DECOMPOSITION: IndicateDeictically(S,H, Location°(e)) $\wedge$
Inform(S,H, Location°(e))

==========

Operatori Multimodali di Maybury (2)

==========

NAME: *Make-entity-visible*
HEADER: MakeVisible(e)
CONSTRAINTS: CartographicEntity(e)
PRECONDITIONS: Displayed(e)
EFFECTS: Visible(e)
DECOMPOSITION: DisplayRegion(e)

==========

NAME: *Explain-route-linguistically-and-visually*
HEADER: ExplainRoute(S, H, from-e, to-e)
CONSTRAINTS: CartographicEntity(from-e) $\wedge$ CartographicEntity(to-e)
PRECONDITIONS: Visible(from-e) $\wedge$ (Goal S KnowHow(H, Go(from-e, to-e)))
EFFECTS: KnowHow(H, Go(from-e, to-e)) $\wedge$
$\forall$segment $\in$ path Know(H,Subpath(segment, path))
DECOMPOSITION: $\forall$segment $\in$ path:
IndicateDeictically(S,H,Source°(segment))
Order(S,H,Go(Source°(segment),
Link°(segment),Destination°(segment))
Identify(S,H,to-e)

Gli operatori possono essere raffinati

- Distinguendo fra precondition 'essenziali' e 'desiderabili':
  i primi devono essere veri perché l'operatore sia applicabile
  i secondi rappresentano vincoli 'deboli'
- Distinguendo fra subgoal 'necessari' ed opzionali'
- E, eventialmente, aggiungendo uno slot 'Rhetorical Relation'.

Operatori Multimodali di Maybury (3)

NAME: *extended-description*
HEADER: Describe (S,H,e)
CONSTRAINTS: Entity (e)
PRECONDITIONS:
  ESSENTIAL: KnowAbout(S,e) $\wedge$ Want(S, KnowAbout(H,e))
  DESIRABLE: $\neg$KnowAbout(H,e)
EFFECTS: KnowAbout(H,e)
DECOMPOSITION: Define(S,H,e)
  optional: Detail(S,H,e) $\wedge$ Divide(S,H,e) $\wedge$
  $\wedge$ Illustrate(S,H,e) $\wedge$ Give-Analogy(S,H,e)

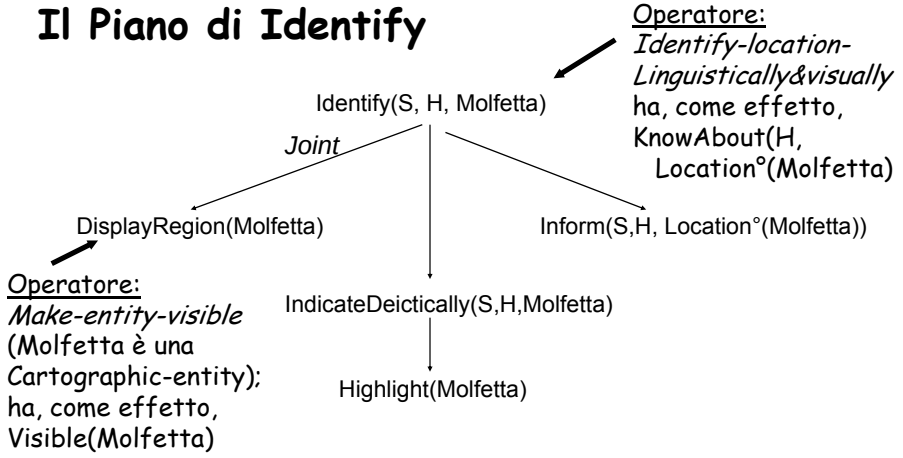
Il goal Describe(S,H,e) si decompone nei sub-goal Define(S,H,e) e (opzionalmente) Detail(S,H,e), Illustrate(S,H,e) e Give-Analogy/S,H,e).

Ha, come precondizioni, che S conosca l'entità da descrivere e che voglia che anche H la conosca. Quest'ultimo è l'effetto prodotto dall'operatore.

Outline di un Algoritmo (semplificato!) di Pianificazione

Considera il Goal g
  Cerca un operatore Op tale che:
    Header(Op) unificabile con g
  Verifica Preconditions(Op);
    $\forall$ Pr: Pr $\in$ Preconditions(Op)
    se Pr non verificata:
      cerca un Op' tale che:
      Effect(Op') unificabile con Pr
      aggiungi Op' al piano
  Verifica Effect(Op)

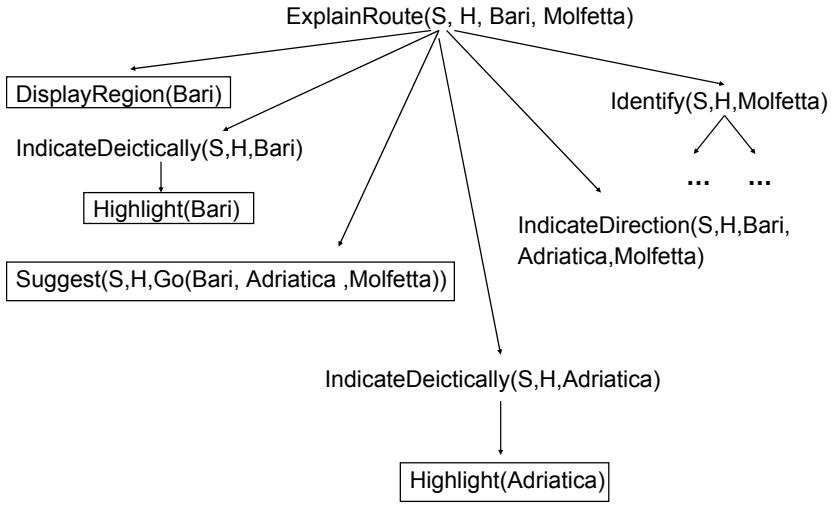
Il Piano di Identify



Operatore:
Identify-location-Linguistically&visually
ha, come effetto,
KnowAbout(H,
Location°(Molfetta)

Operatore:
Make-entity-visible
(Molfetta è una Cartographic-entity);
ha, come effetto,
Visible(Molfetta)

Piano gerarchico per istruzioni multimodali



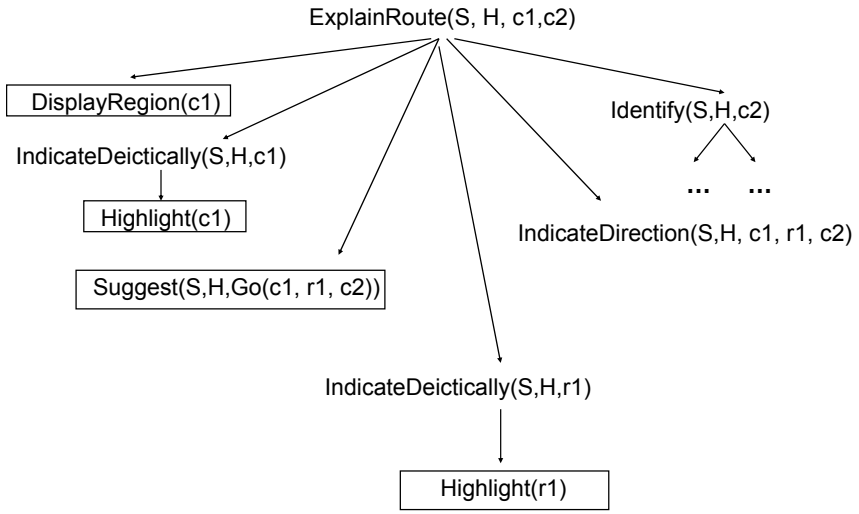
Piani come 'Ricette Preconfezionate'

Quando il messaggio da generare ha una struttura prestabilita, almeno per grandi linee, è possibile rappresentare i piani come 'recipes' non instanziate (strutture ad albero, o anche liste di atti comunicativi, verbali e non verbali), da instanziare di volta in volta con i valori delle variabili che corrispondono al dominio applicativo considerato.

Esempio:

Identify(S,H,e):=
[DisplayRegion(e), Highlight(e), Inform(S,H,Location°(e))]

Piano generico per istruzioni multimodali



I piani possono essere rappresentati come file XML

(che vengono instanziati al momento dell'applicazione)

E' possibile definire un linguaggio xml
per la rappresentazione di 'document plans' (DPML)

Un esempio: il piano generico di Identify(e), con e = 'cartographic-entity':

```
<xml version="1.0">
<DPL>
<goal name="Identify" term="e"  RR="Joint">
        <communicative_act name="DisplayRegion" term="e">
        </communicative_act>
        <goal name="IndicateDeictically" term="e">
                <communicative_act name = "Highlight" term="e">
                </communicative_act>
        </goal>
        <communicative_act name="InformAbout" term="Location°(e)">
        </communicative_act>
</goal>
</DPL>
```

L'attributo RR è facoltativo

I piani devono essere 'raffinati'

prima di essere affidati al 'generatore superficiale'

Un piano conterrà, il più delle volte, *ripetizioni* che renderebbero il discorso pesante e poco naturale.

Il task di *microplanning* ha, come obiettivo, l'*aggregazione* di parti simili.

Esempio:

I ciclamini sono piante da bulbo che fioriscono in inverno.
Le dalie sono piante da bulbo che fioriscono in estate.
Le rose sono arbusti che fioriscono in estate.

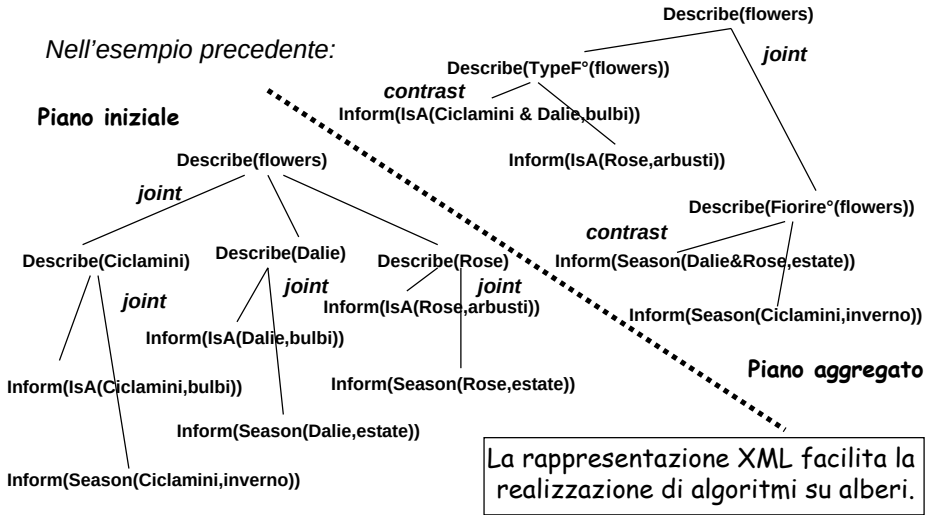
Un'aggregazione di parti simili produrrebbe il testo seguente:

I ciclamini e le dalie sono piante da bulbo, mentre le rose sono arbusti.
I ciclamini fioriscono in inverno, mentre le rose e le dalie fioriscono in estate.

Aggregazione come algoritmo su alberi

La funzione di aggregazione, così come altre procedure di microplanning, trasformano alberi in alberi.

Nell'esempio precedente:



La rappresentazione XML facilita la realizzazione di algoritmi su alberi.

Esercizio

Descrivi il piano dell'esempio precedente con una struttura XML

Realizzazione superficiale

Il piano del discorso è indipendente dalla lingua, dalle caratteristiche dell'apparecchiatura sulla quale verrà realizzato e dalla lingua.

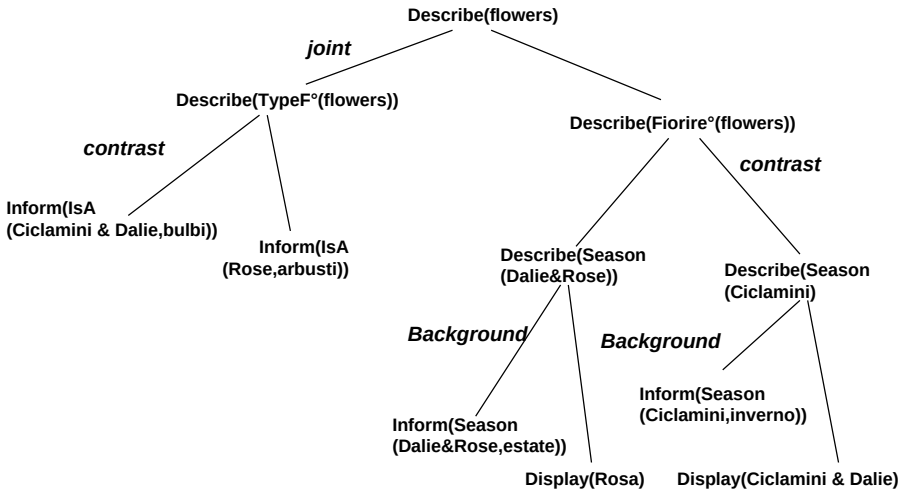
La sua traduzione in un messaggio (in linguaggio naturale, ipermediale, multimediale, in forma vocale o mediante un 'agente animato') viene realizzata in due fasi:

- *generazione della struttura:* si stabilisce l'organizzazione del messaggio in paragrafi o 'pagine', capitoli, sezioni;

- *generazione linguistica*: si 'rende' ogni parte del messaggio nella forma voluta, eventualmente arricchendolo con un 'linguaggio di mark-up' (html o latex).

Riprendiamo l'ultimo esempio

e realizziamolo (in parte) in modo multimediale



Stabilisco dei criteri generali di generazione della struttura

Rendo:

- le *inform* con frasi in linguaggio naturale;
- le *display* con immagini;
- il *focus* del discorso in italic;
- le RR di *contrast* con una disposizione spaziale in cui le due descrizioni vengono messe in sequenza;
- le RR di *background* con una disposizione spaziale in cui la frase e l'immagine sono affiancate (come nell'esempio delle previsioni del tempo di Repubblica),
- ecc

… ottengo il messaggio seguente…

I *ciclamini* e le *dalie* sono piante da bulbo, mentre le *rose* sono arbusti.

Mentre le *dalie* e le *rose* fioriscono in estate,

i *ciclamini* fioriscono in inverno.

Esercizio

Definisci, per grandi linee, un algoritmo che genera, per l'esempio che abbiamo appena visto,
a partire dal piano XML, la pagina html.

Altri criteri

Posso rendere:

- la *sequence* con una lista ordinata,
- la *elaboration process step* con una lista numerata,
- la *contrast* con due colonne di una table,
- …

Ma posso anche rendere le RR con frammenti linguistici

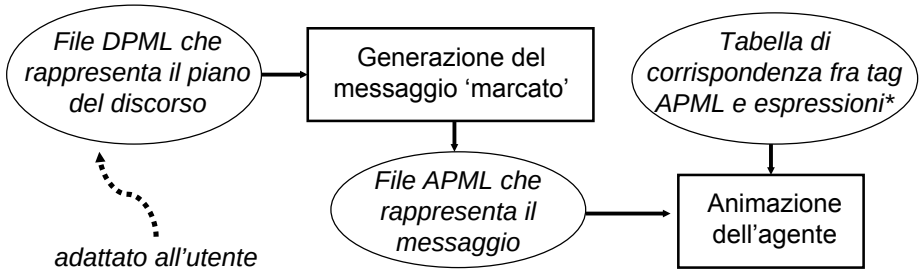
Posso enfatizzare il nucleo oppure il satellite delle *motivation* in bold o con una highlight (per dare rilevanza all'azione suggerita o alla ragione per cui la suggerisco),…

I criteri stabiliti vengono poi tradotti in programmi di generazione del messaggio.

Output come file XML

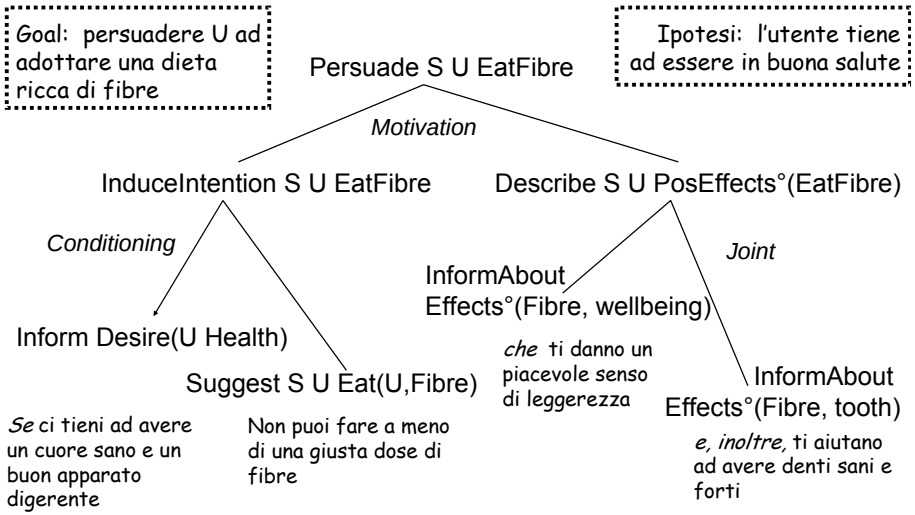
Uno dei casi in cui conviene produrre, da piani del discorso, file marcati è quello della *animazione di agenti conversazionali.*

In questo caso, lo schema di generazione è il seguente:



* "meaning-signal"

Esempio: un messaggio persuasivo a. struttura del piano



Continua l'Esempio: b. rappresentazione DPML

```
<xml version="1.0">
<DPL>
<goal name="Persuade" term="EatFibre"  RR="Motivation">
   <goal name="InduceIntention" term="EatFibre"  RR="Conditioning">
      <communicative_act name="Inform" term="Desire(U,Health)">
      </communicative_act>
      <communicative_act name="Suggest" term="Eat(U Fibre)">
      </communicative_act>
   </goal>
   <goal name="Describe" term="PositiveEffects°(EatFibre) RR="Joint">
      <communicative_act name = "InformAbout" term="Effects°(Fibre,
      wellbeing)"></communicative_act>
      <communicative_act name="InformAbout" term="Effects°(Fibre,
      tooth)"></communicative_act>
   </goal>
</DPL>
```

Continua l'Esempio: c. messaggio come file APML

```
<?xml version="1.0"?>
<APML>
   <performative type="inform">Se ci tieni ad avere <topic-comment
   type="comment">
un cuore sano e un buon apparato digerente, </topic-comment>
non puoi fare a meno di una
<topic-comment type="comment">giusta dose di fibre,
<affective type="happy-for">che ti danno un piacevole senso di
   leggerezza</affective>
</topic-comment>
Le fibre, inoltre,
<topic-comment type="comment">ti aiutano ad avere denti sani e
   forti.</topic-comment>
</performative>
</APML>
```

Continua l'Esempio: d. Tabella di corrispondenza

```
<?xml version="1.0" ?>
<Tagging>
<setup>
<agent name="sally.haptar" />
<volume start="80" /> <speed start="0" /> <voice type="female"/>
</setup>
<performative type="inform">
   <animation /> <TTSsapi5 /> </performative>
<performative type="suggest">- <animation>
   <switchON>M_HeadAside</switchON>
   <switchON>M_Frown</switchON>
   </animation><TTSsapi5 />
</performative>
<affective type="happy-for">-
   <animation><switchON>M_Smile</switchON></animation>
<TTSsapi5 /> </affective> </Tagging>
```

Realizzazione linguistica

Il metodo più semplice consiste nell'associare ad ogni atto linguistico una frase preconfezionata.

Inform(IsA(Ciclamini & Dalie,bulbi)) → *"I ciclamini sono piante da bulbo".*

In questo caso, pero', dovrò memorizzare tante frasi quanti sono gli atti comunicativi da realizzare:
Inform(IsA(Rose,arbusti)) → *"Le rose sono arbusti"*

Anzi, se intendo realizzare messaggi in lingue diverse, dovrò memorizzare una frase per ogni lingua.

Un'alternativa conveniente è quella di definire un algoritmo per la traduzione dell'argomento dell'atto linguistico in (un frammento di) frase in linguaggio naturale:

IsA(Ciclamini & Dalie, bulbi) IsA(Rose,arbusti))
Predicato → verbo: *'sono'*
1° termine → soggetto: *'i ciclamini e le dalie'* *'le rose'*
2° termine → complemento: *'piante da bulbo'* *'arbusti'*

Realizzazione linguistica (continua)

Flourish (Dalie&Rose,estate):
Predicato → verbo: *'fioriscono'*
1° termine → soggetto: *'le dalie e le rose'*
2° termine → complemento: *in estate*

Con questo metodo:
Data una tabella che associa
dd ogni predicato e ad ogni termine un 'frammento di frase', si possono costruire frasi più complesse.

Esempio:
Suggest(S,H,Go(c1, r1, c2))

c1	r1	c2
Bari	Molfetta	Adriatica
Bari	Roma	A1

Conoscenza sul dominio

"Per andare da Bari a Molfetta, ti suggerisco di prendere l'Adriatica.
Per andare da Bari a Roma, è consigliabile prendere l'autostrada A1."

Ma la realizzazione di frasi più complesse richiede un generatore molto più raffinato

Vedremo nella prossima unità i principi che sono alla base della generazione e del riconoscimento di frasi in linguaggio naturale (basati sull'uso di grammatiche).

Segnaliamo, qui, che sono disponibili, su rete, diversi programmi per la generazione di testi in inglese: ad esempio, RealPro (di CoGenTex)

Commento finale: tappe del processo di generazione

Date:

- fonti di conoscenza:
  - **un DB di piani rappresentati come file XML**
  - **una tabella di corrispondenza 'meaning-signal'**
  - **una conoscenza sul dominio**
  - **un modello dell'utente**

- Input:
  - **un goal comunicativo**
- Output:
  - **Un messaggio in linguaggio naturale / un messaggio vocale / un ipertesto / un agente animato che pronuncia il messaggio con espressioni adeguate / ...**

Riferimenti

Building natural language Generation Systems.
E.Reiter e R.Dale.
Cambridge University Press, 2000.

E gli articoli indicati sul sito del corso.

RealPro è descritto su un articolo (breve!) reperibile sul sito del Corso:
http://www.di.uniba.it/people/ArticoliDid/realpro.pdf

Una descrizione completa (con una piccola demo) è visibile su:
http://www.cogentex.com/technology/realpro/index.shtml