Corso di Laurea in
Informatica e Comunicazione Digitale

a.a. 2006 - 2007

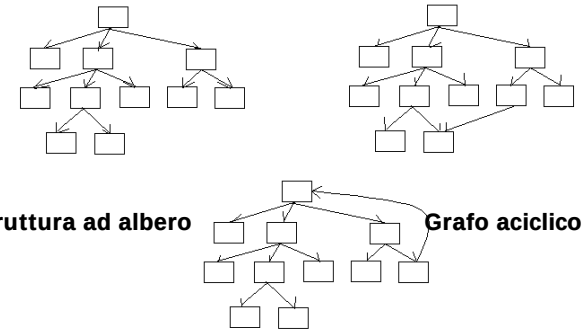
Interazione Uomo-Macchina

Fiorella de Rosis

Programma

- ✰ Introduzione ai metodi per la progettazione di interfacce
- Analisi e caratterizzazione dell'utenza e dei compiti svolti
- Principi di usabilità
- Linguaggi grafici per l'interazione
- ***Metodi formali per la specifica dell'interazione***
- Metodi di valutazione
- Adattamento dell'interazione all'utente e al contesto
- Interazione in linguaggio naturale e multimediale
- Agenti Animati

Ripartiamo dal grafo dei display



Un 'cammino' corrisponde ad una sequenza di passi attraverso i quali si realizza un task

La 'radice' del grafo corrisponde alla finestra di avvio.

Ogni nodo corrisponde ad una nuova finestra

Le foglie sono 'finestre terminali'

Si percorre un arco del grafo effettuando un' azione su un oggetto dell'interfaccia. Si può 'tornare indietro' al nodo-padre chiudendo una finestra aperta.

Cosa permette/non permette questa rappresentazione?

- Ideale per rappresentare la 'navigazione' in un sito ipertestuale o ipermediale: permette di rappresentare lo 'spazio degli stati' e il passaggio da una 'pagina' alla o alle pagine successive.
- Ad ogni nodo può essere associata una singola pagina.
- Ad ogni arco è associata l'azione che l'utente deve eseguire per raggiungere il nodo successivo.

- Nella rappresentazione di interfacce grafiche, questo formalismo presenta vari limiti:
  - Non corrispondenza con la gerarchia dei task;
  - Non rappresentazione delle relazioni fra task;
  - Rappresentazione grossolana degli effetti, sull'interfaccia, di azioni elementari (eventi) effettuate dall'utente.

.... Vediamo quali sono le altrnative...

Perché usare metodi formali per descrivere l'interazione

Come si può descrivere il comportamento dell'interfaccia?
Un esempio da PowerPoint:

Creare una nuova presentazione

Per creare una nuova presentazione è possibile procedere in vari modi. Si può utilizzare la Creazione guidata Contenuto, in cui si inizia a lavorare su una presentazione preimpostata contenente vari suggerimenti riguardo sia al contenuto che alla struttura, oppure si può aprire una presentazione esistente e modificarla in base alle proprie esigenze. Un altro modo per iniziare a creare una presentazione consiste nel selezionare un modello struttura, che definisce la struttura della presentazione, ma non contiene alcun suggerimento riguardo al contenuto. È infine possibile partire da una struttura importata da un'altra applicazione o da una presentazione vuota, che non contiene alcun suggerimento né per il contenuto né per la struttura.

... ambiguità e poca chiarezza...

Utilità dei Metodi Formali

Come per i linguaggi di programmazione, i metodi formali rappresentano un'alternativa per la descrizione non ambigua del linguaggio d'interazione.

Permettono inoltre di effettuare alcune valutazioni di usabilità prima ancora di realizzare un prototipo (e cioè in modo 'pre-empirico').

Metodi di Definizione dei Linguaggi

Come definire in modo non ambiguo il 'linguaggio dei comandi' che l'Utente deve impiegare?

Applicando i metodi sviluppati per definire i linguaggi in generale, fra cui:

- quello naturale e
- i linguaggi di programmazione.

Due classi di metodi:

- metodo *generativo:* le Grammatiche di Reisner
- metodo *riconoscitivo:* le Reti di Petri

5.1 Le Grammatiche Formali

Le Grammatiche formali sono un metodo *generativo* per la definizione dei linguaggi. Descrivono cioè *come generare tutte e sole le stringhe* che appartengono ad un linguaggio.

Grammatica Formale G: $\{\Sigma, V, S, P\}$

alfabeto	Σ: $\{a, b, ...\}$
variabili	V: $\{A, B, ...S\}$
assioma	$S \in V$
produzioni	P: $\{P_i \mid P_i : A \rightarrow \alpha, \quad \alpha \in \{\Sigma \cup V\}$

Un Esempio 'Classico' di Grammatica

Grammatica che genera stringhe nel linguaggio

L: {w | w è una stringa palindroma di lunghezza pari}

Esempio: anna, naaaan, ...

G: {Σ, V, S, P}

Σ: {a, n, λ }

'carattere vuoto'

V: {S}

P: { P_1, P_2, P_3}

P_1 : S → aSa

P_2 : S → nSn

P_3: S → λ

Come genero la stringa 'naan'?

S $\xrightarrow[P_2]{}$ **nSn** $\xrightarrow[P_1]{}$ **naSan** $\xrightarrow[P_3]{}$ **naan**

Linguaggi Context-Free

Linguaggi 'liberi da contesto' (context-free):

nelle produzioni hanno:

- a sinistra, un solo elemento che appartiene a V
- a destra, una stringa qualsiasi composta da elementi di V e di Σ.

Permettono di dare una definizione generativa di un qualsiasi *linguaggio di programmazione*.

Permettono anche di definire i linguaggi di interazione.

Le Grammatiche nei Linguaggi d'Interazione

(P. Reisner, IBM)

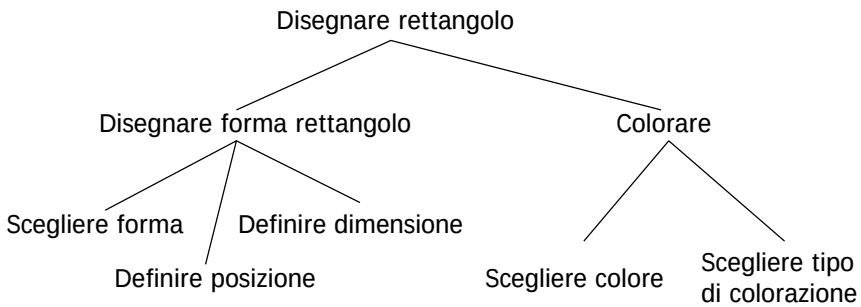
Nelle Grammatiche che sono usate per descrivere i *linguaggi di interazione* fra Utente e Calcolatore:

- *l'alfabeto dei 'terminali'* è dato dall'insieme dei '*task elementari'* che l'Utente può compiere.
- le *variabili* sono i *'task complessi'* (i nodi non foglia nella gerarchia dei task)
- le *produzioni* definiscono *regole di esecuzione dei task complessi*, come combinazione di altri task (elementari o complessi)
- l'*assioma* corrisponde alla variabile che rappresenta la 'radice' dell'albero dei task, e cioè il *task che l'applicazione modellizzata permette di eseguire.*

Un Esempio di Grammatica per Paint:

il Disegno di Rettangoli Colorati

Gerarchia dei task



Un Esempio di Grammatica per Paint:
il Disegno di Rettangoli Colorati

G: {Σ, V, S, P}

S: <disegnare rettangolo>

***Assioma:** radice della Gerarchia dei task*

Σ: {csm(i), point(x, y), psm(x, y),...) }

***Task elementari:** Foglie della gerarchia*

click con il tasto sin del mouse sul control i

puntatore del mouse sulla posizione (x,y)

muovere il mouse, con il tasto sin premuto, sulla posizione (x,y)

V: {<disegnare rettangolo>, <disegnare forma rettangolo>, <scegliere forma>, <definire posizione>, <definire dimensione>, <colorare>, <scegliere colore>, <scegliere tipo colorazione>, ...}

Task complessi

Un Esempio di Grammatica per Paint:
le Produzioni *

<disegnare rettangolo> → <disegnare forma rettangolo>+<colorare>

<disegnare forma rettangolo> → <scegliere forma>+<definire posizione>+<definire dimensione>

<scegliere forma> → *csm(rettangolo)*

<definire posizione> → *point(vertice_sup_sin)*

<definire dimensione> → *psm(vertice_down_dx)*

<colorare>→<scegliere colore>+<scegliere tipo di colorazione>|<scegliere tipo di colorazione>+ <scegliere colore>

<scegliere colore> → *csm(rosso)* | *csm(verde)* | .....

<scegliere tipo di colorazione> → <riempimento uniforme> | <riempimento 'a macchie'> | ...

<riempimento uniforme> → *csm(pennello) + csm(interno-rettangolo)*

...

Il simbolo '+' indica 'concatenazione'

Esercizio 5.1

Definisci una Grammatica per il disegno di ellissi e una Grammatica per il disegno di triangoli in Paint.

Vantaggi delle Grammatiche

- Permettono di descrivere in modo non ambiguo come un task può essere eseguito;
- permettono di valutare la *complessità* di un task, in funzione della 'lunghezza della stringa' in Σ che descrive i comandi elementari attraverso cui il task può essere eseguito;
- permettono di rilevare '*inconsistenze funzionali'* nell'interfaccia di una applicazione.

Un Esempio di Valutazione della Complessità Funzionale

"Disegnare un rettangolo verde al centro dell'immagine";

csm + point + psm + csm + csm + csm

Scegliere forma	Definire posizione	Definire dimensione	Scegliere colore	Scegliere tipo di colorazione	Scegliere forma da colorare

Per ogni 'comando' (o azione elementare),
può essere valutato (in media) il 'tempo necessario per eseguirlo'.
(keystroke-level model)

Questo permette di fare una stima del tempo di esecuzione di task complessi

Complessità funzionale:

Keystroke-level model: Kieras e Polson

Ad ogni azione elementare è associato un tempo di esecuzione

K	pressing a single key or button	~ 40 sec
P	pointing (with a mouse or other device) to a target display	1.1 sec
C	clicking the mouse or a similar device	0.2 sec
H	homing hands on the keyboard or the mouse	0.4 sec
M	mentally prepare to do something	~ 1.4 sec

$$T_f = T_K + T_P + T_C + T_H + T_M$$

Esempio

Inserire la parola 'del' nella frase:
"Programma Corso di Interazione Uomo-Macchina"

- pensare al metodo da applicare
- portare la mano sul mouse
- puntare col mouse prima della parola 'Corso'
- cliccare con il tasto sinistro del mouse
- muovere la mano dal mouse alla tastiera
- pensare alla lettera da battere
- battere tre tasti + lo spazio:

$$T_f = T_M + T_H + T_P + T_C + T_H + T_M + 4T_K$$

Esercizio 5.2

Confronta i diversi modi, in Word, di cancellare una parola di lunghezza n:

- cancellando un carattere alla volta
- isolando la parola col mouse e poi usando il tasto 'cancella'
- altro

Quale dei metodi è più conveniente? In quali condizioni?

Consistenza Funzionale in una Grammatica

Variabili che corrispondono a task simili devono essere generate con regole simili.

Esempio:

In Paint:

<disegnare rettangolo> e <disegnare ovale>

Esercizio 5.3

Verifica, mediante una Grammatica, la consistenza funzionale di due task simili in Paint.

Limiti delle Grammatiche

Non permettono di rappresentare:

- i casi in cui l'ordine di esecuzione di una serie di task non è rigidamente definito (o richiedono un numero elevato di produzioni, per farlo)

Esempio: **in un disegno complesso, posso disegnare le varie componenti in un ordine qualsiasi.**

- la dinamica con cui l'interfaccia evolve in relazione ai comandi inviati dall'Utente al Sistema

  Esempio: csm(rettangolo) produce un cambiamento nell'icona e l'apertura di nuovi control

- l'adattamento dell'interfaccia all'Utente

Limiti della Gerarchia dei Task - delle Grammatiche

I limiti che abbiamo rilevato per la definizione 'Generativa' dei linguaggi di interazione

sono simili a quelli che avevamo rilevato, nell'Unità 1, per la rappresentazione della decomposizione dei task mediante gerarchie.

D'altra parte, abbiamo visto che il grafo dei display non permette di rappresentare in modo soddisfacente gli aspetti dinamici dell'interazione.

Come superare questi limiti?

User Action Notation (per i task)

Reti di Petri (per i linguaggi)

5.2 Metodi Riconoscitivi:
le Reti di Petri (PN)

Definizione 'formale'

Una PN è una *tupla:* <N, P, T, A, SP, EP, M>, con

N: *nome* della rete
P: un insieme di *posti,* indicati con ○
T: un insieme di *transizioni*, indicati con ▭
A: un insieme di archi orientati fra P e T (o T e P): ⟶
A(p,t) = 0/1; A(t,p) = 0/1
SP: un insieme di posti (incluso in P) denotati come '*start place'*
EP: un insieme di posti (incluso in P) denotati come '*end place'*

(segue)

5.2 Metodi Riconoscitivi:
le Reti di Petri (PN)

Definizione 'formale' (segue)

M: una *funzione di marcatura* dei posti $M(p_i)$, che associa ad ogni posto il valore di una variabile binaria (a seconda che il posto sia 'marcato' o no).

Un posto p_i marcato viene indicato con ◎

In questo caso, $M(p_i) = 1$

Uso delle Reti di Petri
per la Simulazione del Comportamento di Interfacce

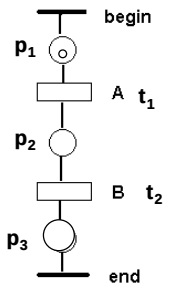
Definisco:

- *Preset di una transizione t:*
  l'insieme dei posti p che *precedono* la transizione (A(p,t) = 1);
- *Postset di una transizione t:*
  l'insieme dei posti p che *seguono* la transizione (A(t,p) = 1)

All'inizio della simulazione, tutti i posti in SP sono marcati.

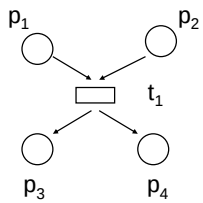
La *funzione M di propagazione delle marcature* è definita così:
una transizione può essere *attivata* se e solo se tutti i posti nel suo preset sono marcati;
quando una transizione viene attivata, tutte le marcature passano dal suo preset al suo postset.

Qualche Esempio di Uso delle PN



SP = $\{p_1\}$
EP = $\{p_3\}$

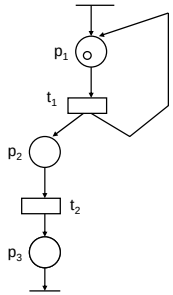
Inizialmente:
p_1 è marcato; t_1 può essere attivata;
quando viene attivata t_1 ; la marcatura si sposta su p_2 ;
t_2 può, ora, essere attivata;
quando viene attivata t_2 ,
la marcatura si sposta su p_3 e il task è completato.



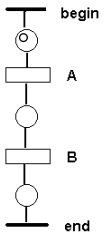
t_1 può essere attivata soltanto quando sia p_1 che p_2 sono marcati;
quando t_1 viene attivata, sia p_3 che p_4 vengono marcati.

Esercizio 5.4

Come vengono propagate le marcature, in questa PN?

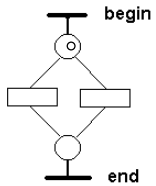


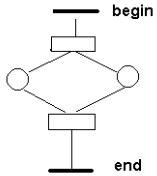
Costrutti Elementari nelle PN



Sequenza: **le transizioni hanno preset e postset diversi**

Scelta: **due o più transizioni hanno il preset in comune**

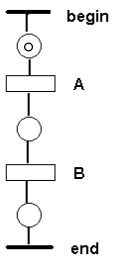




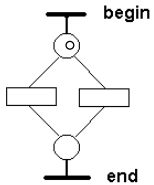
Concorrenza: **una transizione ha un postset che include diversi elementi**

Corrispondenza fra UAN e Reti di Petri

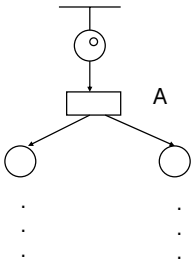
Sequenza
AB



Scelta
A|B

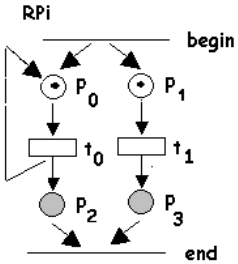


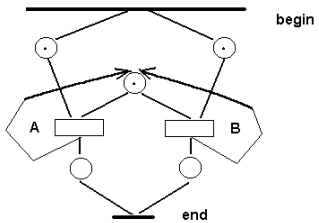
Concorrenza visiva



Corrispondenza fra UAN e Reti di Petri

Iterazione
(A)*||B

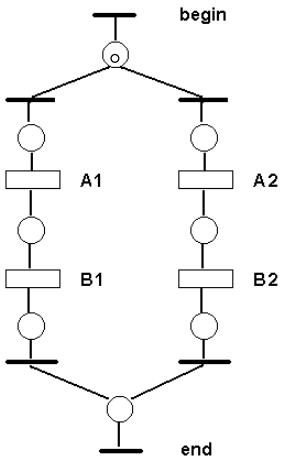




Indipendenza d'ordine
A&B

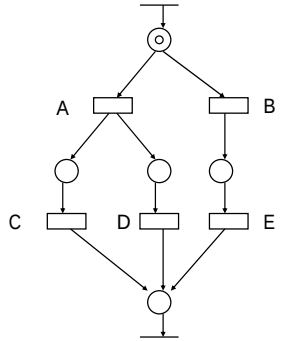
I Costrutti Elementari possono essere Combinati per Formare Reti più Complesse

(A1 A2)|(B1 B2)



Esercizio 5.5

Disegna la UAN che corrisponde a questa PN



Combinazione di sequenza, scelta e concorrenza visiva

Estensione delle PN per la Rappresentazione dell'Interazione (XDM)

Ad ogni posto ◯ viene associata la descrizione di uno *stato del sistema,* rappresentata mediante un oggetto grafico semplice o complesso

Una singola icona

Un'intera finestra

Una toolbar o un gruppo di icone

Ecc…

Quando, nella PN che simula l'interazione in una applicazione, il posto viene 'marcato', l'oggetto grafico associato viene mostrato sul display

Le PN per la Rappresentazione dell'Interazione (continua)

Ad ogni transizione viene associata una *azione* compiuta dall'utente (un *evento*):

csm: click su un oggetto con il tasto sin del mouse

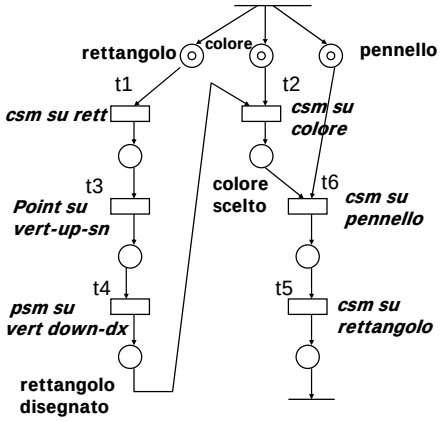
text: inserimento di una stringa da tastiera

menu: scelta di un item da un menu

.... Eccetera

Quando una transizione viene attivata, gli oggetti associati al suo preset spariscono dal display e quelli associati al suo postset vengono visualizzati.

Un Esempio di PN per Paint



'Disegno il rettangolo' deve essere completato prima di 'coloro il rettangolo'.

La transizione 'csm su colore' è attivabile soltanto dopo che il posto 'rettangolo disegnato' è stato marcato.

La transizione 'csm su pennello' è attivabile soltanto dopo che il posto 'colore scelto' è stato marcato.

Esercizio

Secondo la PN precedente, per colorare un rettangolo bisogna *prima* scegliere il colore e *poi* il tipo di colorazione, mentre cosi' non è.

Prova a modificare la rete per rappresentare l'indipendenza d'ordine fra i due subtask e il fatto che, se si utilizza il colore di default, non occorre selezionarlo.

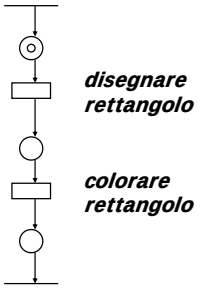
Innestamento delle Transizioni e Rappresentazione Top-Down dei Task

Il modello formale di un'interfaccia può essere costruito in modo top-down, riproducendo la gerarchia dei task.

Vengono introdotte delle 'transizioni innestate', rappresentate con 'rettangoli scuri', a cui possono essere associati task complessi.

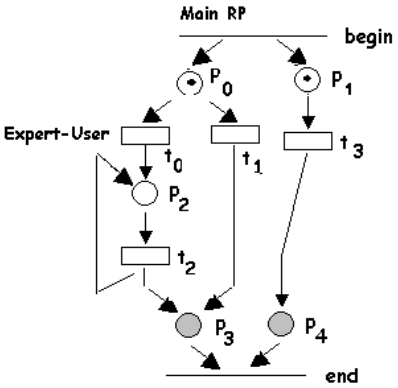
Un Esempio di PN con Transizioni Innestate

PN name: 'disegnare un rettangolo colorato'



... e poi sviluppo i task di 'disegnare rettangolo' e 'colorare rettangolo' in PN più semplici

Esercizio 5.6



Traduci questa rete
in una rete
con innestamento

Esercizio 5.7

Rappresenta, con una rete innestata, il task di compilazione di una email in uno dei software che conosci

Inserimento di Condizioni nelle PN

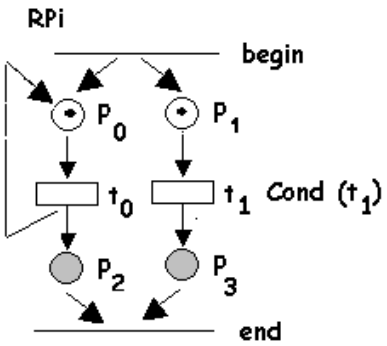
- Condizioni associate ai 'posti':

Cond(p_i) ***p_i può venire marcata soltanto se la condizione è verificata***

- Condizioni associate alle 'transizioni'

Cond(t_j) ***t_j può venire attivata soltanto se la condizione è verificata***

Esempi Di PN Con Condizioni

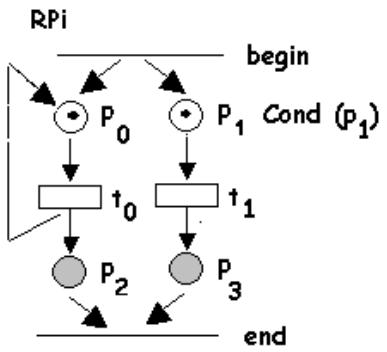


Condizionamento
di transizioni

Se la condizione è verificata, entrambi i task sono eseguibili.

Altrimenti è eseguibile soltanto t0

Esempi Di PN Con Condizioni



Condizionamento di posti

Il posto p1 viene marcato soltanto se la condizione è verificata (e, di conseguenza, t1 è eseguibile).

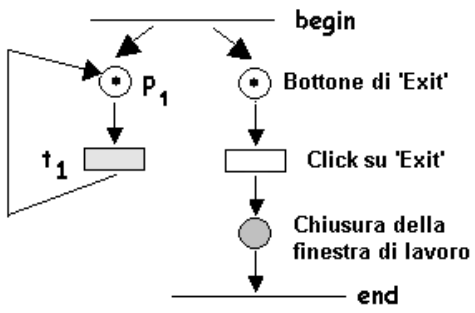
Altrimenti è eseguibile soltanto t0

Esercizio

Rispetto alla modalità di esecuzione dei task rappresentati, le reti mostrate nei due lucidi precedenti sembrano essere identiche.

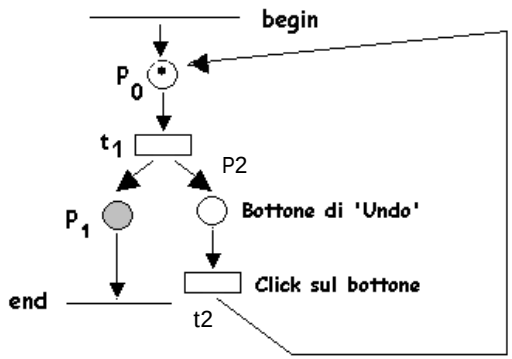
Rifletti però su qual'è la differenza fra i comportamenti dell'interfaccia descritti dalle due reti.

Esempi di PN per Funzioni Comuni



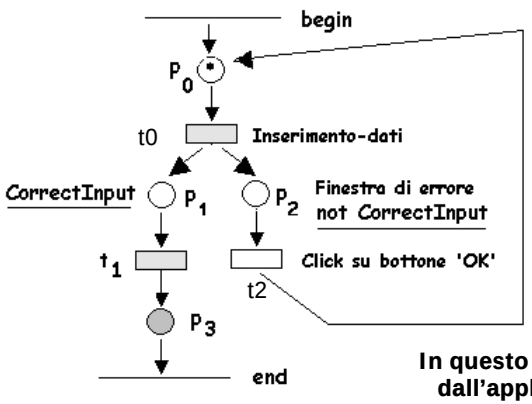
La funzione di Exit

Esempi di PN per Funzioni Comuni



La funzione di 'undo'

Esempi di PN per Funzioni Comuni



Segnalazione di errore

Nota:
In questo caso, non posso uscire dall'applicazione finché non ho eseguito correttamente il task associato a t0

Descriviamo il funzionamento della rete precedente

La rete mostra che, inizialmente, è marcato il posto p0 (il solo che appartiene all'insieme SP).

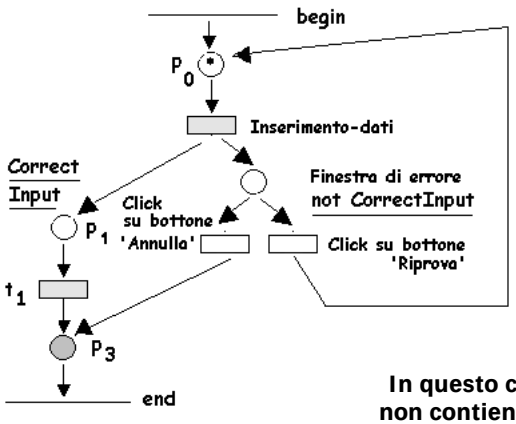
L'unico task eseguibile è quindi quello di inserimento dati, associato alla transizione t0.

Quando questo task viene eseguito, possono verificarsi due situazioni:

- Se l'input era corretto, il task associato alla transizione t1 diventa eseguibile; se lo si esegue, si raggionge l'EP p3;
- Se l'input conteneva un errore, si apre una finestra di errore che contiene un bottone di 'OK', cliccando sul quale è possibile riprendere il task dall'inizio.

Nota: descrizione da effettuare nella Relazione…!!

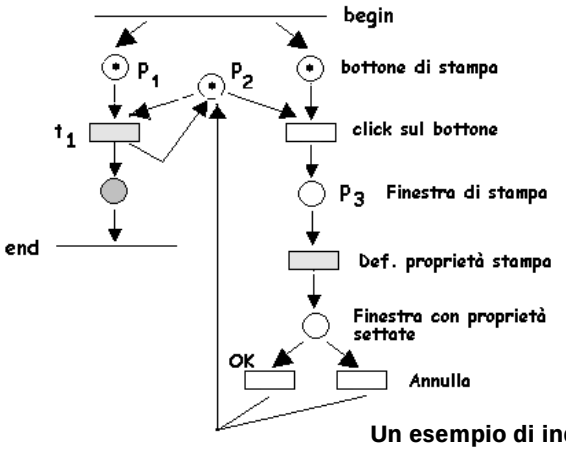
Esempi di PN per Funzioni Comuni



Segnalazione di errore con possibilità di uscire

Nota:
In questo caso, la finestra di errore non contiene un bottone di 'OK', ma due bottoni, di 'Annulla' e 'Riprova'

Esempi di PN per Funzioni Comuni



Funzione di stampa

Nota:
Un esempio di indipendenza d'ordine, con uno dei due task opzionale

Esercizio 5.8

Rappresenta, mediante il condizionamento di posti oppure mediante il condizionamento di transizioni, i seguenti aspetti dell'interazione:

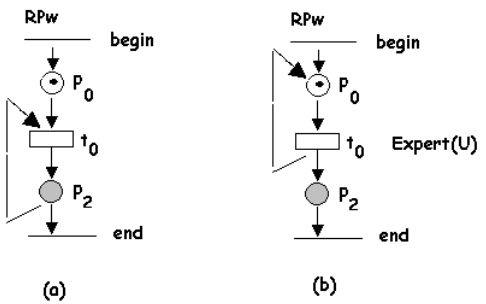
- Attivazione condizionata del control 'taglia' in PowerPoint
- Divieto di accesso alla rete, con messaggio di errore, in caso di connessione interrotta

Quali Valutazioni Possono Essere Effettuate, con le PN?

- *completezza*
- *rispetto dei vincoli di accesso*
- *consistenza*
- *complessità funzionale*

a. Verifica di Completezza

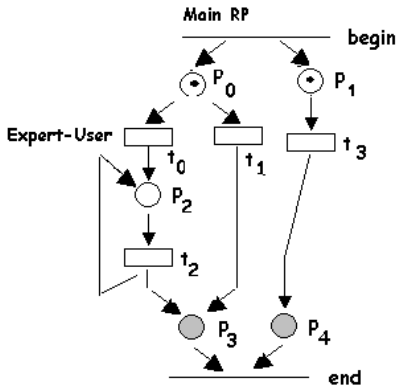
Dato un task T e la Rete di Petri RPt che lo rappresenta, la verifica della possibilità di effettuare il task T si traduce nella verifica della *raggiungibilità di almeno uno dei posti nell'insieme degli end place EP di RPt.*



Un esempio: il task rappresentato nella rete (b) non è effettuabile per un utente non esperto.

b. Verifica di rispetto dei vincoli di accesso

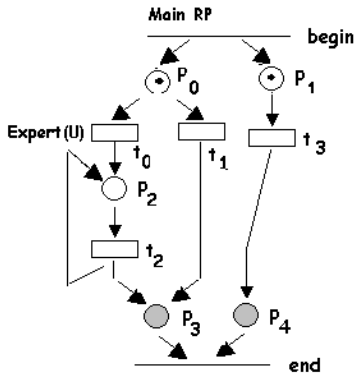
Una interfaccia rispetta i vincoli di accesso stabiliti per le diverse categorie di utenti se permette di effettuare solo i task consentiti per ogni categoria.



Un esempio:

Vincolo d'accesso a t0, t2 sull'esperienza dell'utente

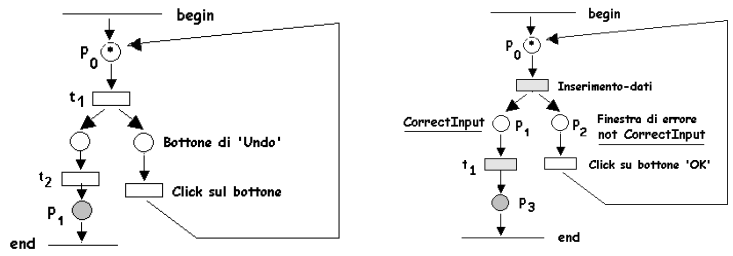
Esercizio 5.9



Descrivi, in linguaggio naturale, il funzionamento di questa rete per gli utenti esperti e non esperti.

c. Verifica di consistenza funzionale

Dati due task 'simili' T1 e T2 e le Reti di Petri RPt1 e RPt2 che li rappresentano, la verifica di consistenza funzionale di T1 e T2 si traduce nella verifica della *somiglianza dei cammini che legano start place a end place* in RPt1 e RPt2.



Esempio: le RP che rappresentano i task di 'undo' e di 'segnalazione di errore' sono simili: i due task che le reti descrivono sarebbero quindi realizzati in modo consistente.

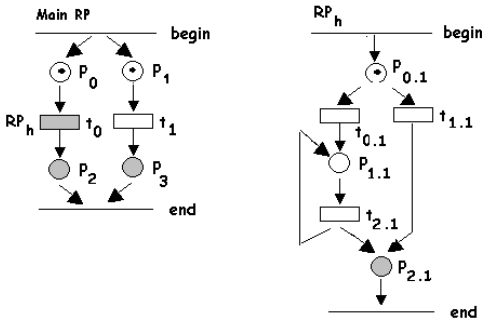
Esercizio 5.10

Rappresenta in una RP il disegno di rettangoli e di ellissi in Paint e discuti la consistenza dei due task.

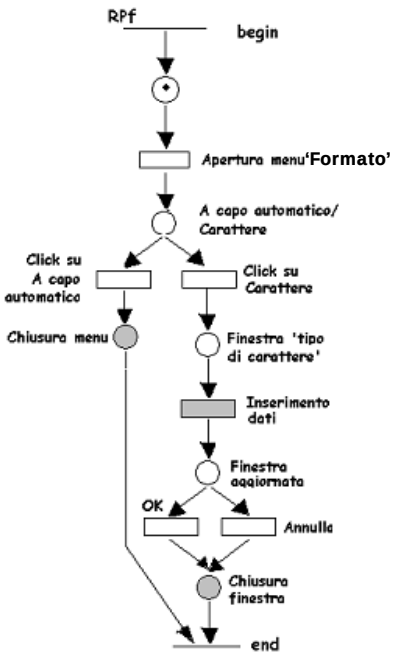
d. Verifica di complessità funzionale

La complessità di un task t rappresentato da una RPt è una funzione lineare della lunghezza del cammino che lega SP a EP in RPt.

Se questo cammino non è unico, è possible confrontare (in termini di complessità) i modi alternativi di compiere il task.



Il task rappresentato in questa RP può essere eseguito nei tre modi seguenti: t_{01} $(t_{21})^*$ oppure t_{11} oppure $t_{1.}$ Il tempo di esecuzione di ciascuno di essi può essere stimato, come per le Grammatiche, con il metodo di Kieras e Polson.



Esercizio 5.11

Calcola la complessità funzionale del task 'scelta del formato' in blocco note, rappresentato in questa figura

Per Approfondimenti

Sulle Reti di Petri:

F de Rosis, S Pizzutilo e B De Carolis:
Formal description and evaluation of user-adapted interfaces.
International Journal of Human-Computer Studies, 49, 1998

Due pagine web sui tool disponibili:
http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/
http://it.wikipedia.org/wiki/Lista_dei_tools_per_le_reti_di_Petri

DaNAMiCS, un tool free in Java della University di Cape Town:
http://www.cs.uct.ac.za/Research/DNA/DaNAMiCS/DaNAMiCS.html