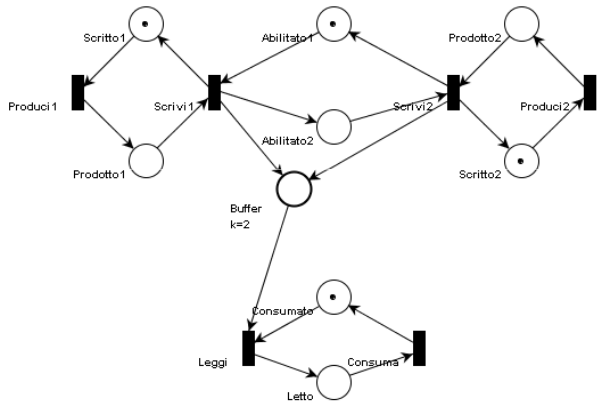
Esercizi Esame

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Esercizio 1

- Un sistema software è costituito da due processi produttori, identici, che scrivono messaggi su un buffer capace di memorizzare sino a due messaggi. I processi produttori devono alternarsi nella scrittura, cioè ognuno dei due, dopo aver scritto un messaggio, deve attendere che l'altro processo scriva a sua volta, prima di scriverne un altro. I messaggi sono letti da un processo consumatore. Inizialmente i produttori sono pronti a produrre e il consumatore pronto a leggere.
- Specificare il sistema con una rete di Petri.

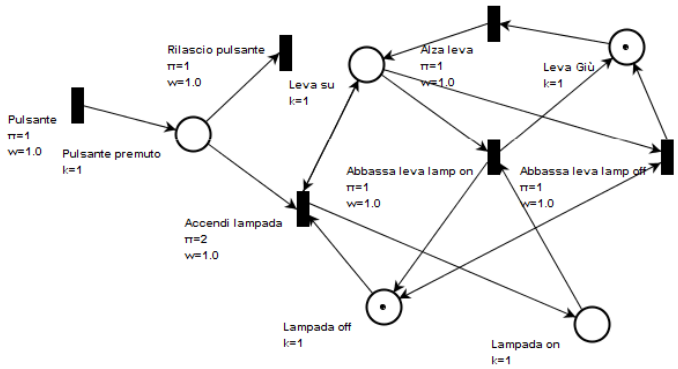
Soluzione



Esercizio 2

- Specificare mediante una rete di Petri il funzionamento di sistema composto da un interruttore a pressione, una leva a due posizioni (0,1) e una lampadina. La lampada, inizialmente spenta, si accende quando viene premuto il pulsante e la leva si trova in posizione 1. L'accensione della lampada lascia la leva nel suo stato. La lampada si spegne quando la leva viene portata in posizione 0.

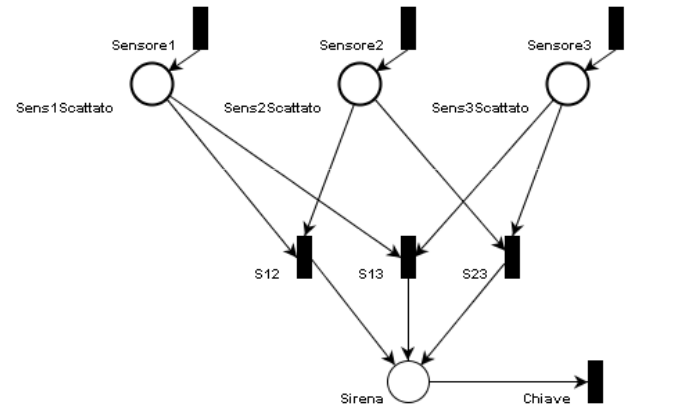
Soluzione



Esercizio 3

- Specificare mediante una rete di Petri il funzionamento di un sistema di allarme di un edificio dotato di una sirena, tre sensori identici che rilevano presenze all'interno dell'edificio e una chiave di disinserimento dell'allarme. La sirena si accende quando almeno due dei tre sensori rilevano una presenza. Dopo che è scattata la sirena, questa viene spenta se viene girata la chiave.

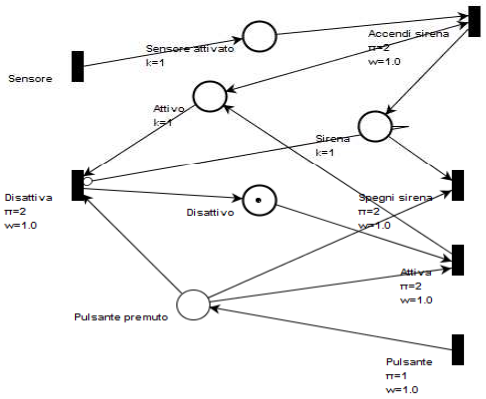
Soluzione



Esercizio 4

- Specificare mediante una rete di Petri il funzionamento di un sistema di allarme, dotato di un pulsante di attivazione/disattivazione, un sensore acustico e una sirena. Il sistema di controllo si attiva premendo il pulsante. A sistema attivato, se il sensore acustico rileva un rumore si attiva la sirena. La sirena si spegne quando viene nuovamente premuto il pulsante. La pressione del pulsante a sirena accesa non disinserisce il sistema di controllo. Il sistema di controllo viene disinserito, da inserito, quando viene premuto il pulsante a sirena spenta.

Soluzione



Esercizio 5

- Si dia una specifica in Z di un videonoleggio. Ciascun cliente è in possesso di un tesserino magnetico con un codice identificativo unico (intero positivo). Ciascuna videocassetta è a sua volta identificata da un numero intero positivo e ha associato un titolo (stringa di 20 caratteri) e un numero di copie.
- Si chiede di specificare le seguenti operazioni in Z:
  - inserimento di una nuova videocassetta, specificando in ingresso titolo, codice e numero di copie: l'operazione va a buon fine solo se non esiste già in catalogo una videocassetta con quel codice;
  - aggiunta di un certo numero di copie (N) a una videocassetta (con codice dato in ingresso) già in catalogo: l'operazione va a buon fine solo se esiste già in catalogo una videocassetta con quel codice;
  - noleggio di una videocassetta (con codice dato in ingresso) da parte di un cliente identificato tramite tesserino magnetico: tale operazione va a buon fine solo se esiste almeno una copia della videocassetta richiesta;
  - restituzione di una videocassetta (con codice dato in ingresso) da parte di un cliente identificato tramite tesserino magnetico.

Tipi, variabili

- Tipi definiti dall'utente
  [Titoli,Clienti]
  - Titoli = insieme dei titoli delle videocassette
  - Clienti = insieme dei codici dei clienti
- Variabili che descrivono lo stato del sistema:
  - catalogo: funzione parziale che associa ad un codice il titolo della videocassetta
  - copie: funzione parziale che associa ad un codice il numero di copie della videocassetta
  - noleggi: relazione che associa ai clienti i codici delle videocassette da loro noleggiate

Stato, inizializzazione, successo

Noleggio

$catalogo: \mathbb{N}_1 \rightarrow Titoli$
$copie: \mathbb{N}_1 \rightarrow \mathbb{N}_1$
$noleggi: \mathbb{N}_1 \leftrightarrow \mathbb{N}_1$

$\forall vc: dom(catalogo) \bullet$
$|noleggi^{-1}(vc)| \leqslant copie(vc)$

InitNoleggio

Noleggio'

$catalogo' = \emptyset$
$copie' = \emptyset$
$noleggi' = \emptyset$

Successo

$rep!: Report$

$rep! = 'Okay'$

Inserimento di una videocassetta

Precondizioni:
non esiste una videocassetta con quel codice

InserisciVideoCassetta

ΔNoleggio
titolo?: Titoli
codice?: $\mathbb{N}_1$
n_copie?: $\mathbb{N}_1$

codice ∉ dom catalogo
catalogo'=catalogo ∪ {codice?↦ titolo?}
copie'= copie ∪ {codice?↦ n_copie?}
noleggi'=noleggi

CodicePresente

ΞNoleggio
codice?: $\mathbb{R}$
rep!: Report

codice? ∈ dom catalogo
rep! = 'Codice già presente'

$$Inserimento \triangleq InserisciVideoCassetta \wedge Successo \vee CodicePresente$$

Aggiunta copie

Precondizioni:
videocassetta già in catalogo

AggiuntaCopie

ΔNoleggio
codice?: $\mathbb{N}_1$
N?: $\mathbb{N}_1$

codice? ∈ dom catalogo
catalogo'=catalogo
copie'= copie ⊕ {codice? ↦ (copie(codice?)+N?)}
noleggi'=noleggi

CodiceAssente

ΞNoleggio
codice?: $\mathbb{R}$
rep!: Report

codice? ∉ dom catalogo
rep! = 'Codice assente'

$$Aggiunta \triangleq AggiuntaCopie \wedge Successo \vee CodiceAssente$$

Noleggio di una videocassetta

Precondizioni:
esiste una copia disponibile

RichiestaNoleggioVC

ΔNoleggio
codice?: $\mathbb{N}_1$
cliente?: $\mathbb{N}_1$

$|noleggi^{-1}(codice?)| < copie(codice?)$
noleggi'=noleggi ∪ {cliente? ↦ codice?}
catalogo'= catalogo
copie'=copie

NoCopia

ΔNoleggio
codice?: $\mathbb{N}_1$
rep!: Report

$|noleggi^{-1}(codice?)| = copie(codice?)$
rep! = 'Nessuna copia disponibile'

$$NoleggioVC \triangleq RichiestaNoleggioVC \wedge Successo \vee NoCopia$$

Restituzione di una videocassetta

Precondizioni: nessuna

RestituzioneVC

ΔNoleggio
codice?: $\mathbb{N}_1$
cliente?: $\mathbb{N}_1$

noleggi'=noleggi \{cliente? ↦ codice?}
catalogo'= catalogo
copie'=copie

$$Restituzione \triangleq RestituzioneVC \wedge Successo$$

Esercizio 7

- Si vuole specificare in Z la gestione di prestiti delle videocassette di un videonoleggio ai suoi clienti (registrati). Ciascun cliente è caratterizzato dal proprio codice fiscale (stringa di 16 caratteri) e da un codice cliente che gli viene assegnato all'atto della registrazione. Ciascun utente ha un conto aperto con deposito iniziale di 10 euro. Il prestito di una videocassetta può avere luogo solo se ci sono ancora soldi disponibili in conto.
- Il videonoleggio ha a disposizione un catalogo di videocassette, ciascuna caratterizzata da un codice. Si suppone che ogni opera sia in copia unica. Ogni videocassetta può essere disponibile, oppure in prestito. Per ciascun prestito il costo è di 1,5 euro (si suppone la durata del prestito sia fissa).

Esercizio 7

- Si modellino le seguenti operazioni:
  - Registrazione di un nuovo cliente: dato il codice fiscale del cliente, viene assegnato un nuovo codice identificativo, si aggiunge l'utente all'elenco di quelli già registrati e gli si assegna il deposito iniziale di 10 euro che viene pagato seduta stante dal cliente; se il codice fiscale risulta tra quelli dei clienti già registrati si dà una segnalazione di errore;
  - Richiesta di un prestito: dati il codice cliente e il numero di inventario di una videocassetta, il sistema controlla che esista un cliente registrato con quel codice e una videocassetta con quel numero di inventario disponibile. Se il cliente ha ancora disponibilità in conto superiore o uguale al costo del noleggio, si modifica lo stato della videocassetta da disponibile a "in prestito" e si addebita il costo al cliente decrementando il suo deposito di 1,5 euro;
  - Restituzione di una videocassetta: dati il codice cliente e il numero di inventario di una videocassetta, il sistema controlla che esista un cliente registrato con quel codice e una videocassetta con quel numero di inventario "in prestito". In questo caso modifica lo stato della videocassetta da "in prestito" a disponibile.

Tipi, variabili

- Tipi definiti dall'utente
  [CodiceFiscale,CodiciInventario]
  - CodiciFiscali= insieme dei codici fiscali delle persone
  - CodiciInventario= codici di inventario delle videocassette
- Variabili che descrivono lo stato del sistema:
  - clienti: funzione parziale che associa ad un codice fiscale il codice cliente. Il codice cliente è un intero positivo.
  - prestiti: funzione totale che associa al codice di inventario delle videocassette il codice del cliente che l'ha in prestito oppure 0 se è disponibile.
  - crediti: funzione parziale che associa ad un intero positivo il credito residuo del cliente con quel codice in centesimi di euro

Stato, inizializzazione, successo

Noleggio

$clienti: CodiciFiscali \pfun \mathbb{N}_1$

$prestiti: CodiciInventario \fun \mathbb{N}_1$

$crediti: \mathbb{N}_1 \pfun \mathbb{N}$

InitNoleggio

$Noleggio'$

$clienti' = \emptyset$

$prestiti' = \{ cassetta : CodiciInventario \mid cassetta \mapsto 0 \}$

$crediti' = \emptyset$

Successo

$rep! : Report$

$rep! = 'Okay'$

Registrazione di un nuovo cliente

Precondizioni:
il cliente non deve già essere registrato

RegistrazioneNuovoCliente

ΔNoleggio
cliente?: CodiciFiscali
codice!: $\mathbb{N}_1$

cliente? $\notin$ dom clienti
codice! $\notin$ ran clienti
clienti'=clienti $\cup$ {cliente? $\mapsto$ codice!}
prestiti' = presititi
crediti' = credito $\cup$ {codice! $\mapsto$ 1000}

ClienteGiàRegistrato

ΞNoleggio
rep!: Report

cliente? $\in$ dom clienti
rep! = 'Cliente già registrato'

Registrazione $\hat{=}$ RegistrazioneNuovoCliente $\wedge$ Successo
$\vee$
ClienteGiàRegistrato

Richiesta di prestito

Precondizioni:
1) l'utente deve essere registrato
2) la videocassetta deve essere disponibile
3) il credito del cliente è superiore a 150 centesimi di euro

RichiestaPrestito

ΔNoleggio
cliente?: $\mathbb{N}_1$
cassetta?: CodiciInventario

cliente? $\in$ ran clienti
prestiti(cassetta?) $-$ 0
crediti(cliente?) $\geq$ 150
clienti' = clienti
prestiti' = presititi $\oplus$ {cassetta? $\mapsto$ cliente?}
crediti' = crediti $\oplus$ {cliente? $\mapsto$ crediti(cliente?)-150}

Richiesta di prestito

ClienteNonRegistrato

ΞNoleggio
cliente?: $\mathbb{N}_1$
rep?: Report

cliente? $\notin$ ran clienti
rep?='Il cliente non è registrato'

CassettaNonDisponibile

ΞNoleggio
cassetta?: CodiciInventario
rep?: Report

prestiti(cassetta?) $\neq$ 0
rep?='La cassetta non è disponibile'

CreditoInsufficiente

ΞNoleggio
cliente?: $\mathbb{N}_1$
rep?: Report

crediti(cliente?) < 150
rep?='Credito insufficiente'

Prestito $\hat{=}$ RichiestaPrestito $\wedge$ Successo
$\vee$
ClienteNonRegistrato
$\vee$
CassettaNonDisponibile
$\vee$
CreditoInsufficiente

Restituzione di una videocassetta

Precondizioni:
La videocassetta deve essere in prestito

RestituzioneCassetta

ΔNoleggio
cliente?: $\mathbb{N}_1$
cassetta?: CodiciInventario

cliente? $\in$ ran clienti
prestiti(cassetta?) $\neq$ 0
clienti' = clienti
prestiti' = presititi $\oplus$ {cassetta? $\mapsto$ 0}
crediti' = crediti

CassettaNonInPrestito

ΞNoleggio
cassetta?: CodiciInventario
rep?: Report

prestiti(cassetta?) = 0
rep?='La cassetta non in prestito'

Restituzione $\hat{=}$ RestituzioneCassetta $\wedge$ Successo
$\vee$
ClienteNonRegistrato
$\vee$
CassettaNonInPrestito

Esercizio 8

- Un laboratorio analisi di un ospedale ha quattro macchinari identici, ciascuno in grado di eseguire venti diversi tipi di analisi (identificate con un numero da 1 a 20). Le analisi vengono richieste dai medici dei vari reparti dell'ospedale. Ciascun medico è identificato da un codice, assegnato all'atto dell'assunzione del medico. I macchinari sono disponibili dalle 8 del mattino sino alle 19 e si suppone che ciascuna analisi richiesta duri un'ora.
- Quando un medico prenota un'analisi, deve specificare il proprio codice e il tipo di analisi (da 1 a 20). Accertato che il medico appartenga all'ospedale, gli viene assegnato un orario (il più presto possibile) e un macchinario. Non c'e' limite al numero di richieste che un medico può fare nel corso della giornata. Un macchinario non può svolgere più di un'analisi allo stesso tempo.
- La prenotazione va a buon fine solo se esiste un macchinario libero su cui svolgere l'analisi.
- Per semplicità si ipotizzi che:
  - Le prenotazioni possano essere effettuate solo per il giorno corrente;
  - L'ora sia individuata da un intero (compreso fra 8 e 18).
- Si specifichi in Z un tale sistema di gestione delle prenotazioni delle analisi.

Tipi, variabili

- Tipi definiti dall'utente
  [Tipo, Ora, Macchina]
  - Tipo=1..20
  - Ora=1..18
  - Macchina=1..4
- Variabili che descrivono lo stato del sistema:
  - medici: insieme dei medici dell'ospedale, i medici sono identificati da numeri interi non negativi
  - prenotazioni: funzione parziale che associa alle ore e alle macchine il medico che le ha prenotate e il tipo dell'analisi

Stato, inizializzazione, successo

Laboratorio

$medici: \mathbb{P}\,\mathbb{N}$

$prenotazioni: Ora \times Macchina \pfun \mathbb{N} \times Tipo$

$medici \supseteq \{medico \mid (\exists\, tipo \bullet (medico, tipo) \in ran\ prenotazioni)\}$

InitLaboratorio

$\Delta Laboratorio$

$prenotazioni' = \varnothing$

$medici' = \varnothing$

Successo

$rep!: Report$

$rep! = 'Okay'$

Prenota analisi

Prenota

$\Delta Laboratorio$

$codice\_medico?: \mathbb{N}$

$tipo?: Tipo$

$ora!: Ora$

$macchina!: Macchina$

$codice\_medico? \in medici$

$(ora!, macchina!) \notin dom\ prenotazioni$

$\exists\, ora1: Ora, macchina1: Macchina \bullet$

$(\ (ora1, macchina1) \notin dom\ prenotazioni \land$

$\quad ora1 < ora!\ \ )$

$prenotazioni' = prenotazioni \cup \{(ora!, macchina!) \mapsto (codice\_medico?, tipo?)\}$

Precondizioni:
1) il medico appartiene all'ospedale
2) c'e' una macchina libera

Prenota analisi

MedicoNonAppartenente

ΞLaboratorio
codice_medico?: $\mathbb{N}$
rep!: Report

codice_medico? $\notin$ medici
rep! = 'Il medico non fà parte dell'ospedale'

NessunaMacchinaLibera

ΞLaboratorio
rep!: Report

$\nexists$ ora: Ora, macchina: Macchina •
((ora,macchina) $\notin$ dom prenotazioni)
rep! = 'Nessuna macchina libera'

PrenotaAnalisi $\widehat{=}$ Prenota $\wedge$ Successo
$\vee$
MedicoNonAppartenente
$\vee$
NessunaMacchinaLibera

Esercizio 9

- Si vuole specificare in Z la procedura di controllo dei depositi bilanci societari presso gli sportelli del tribunale di Bologna. Ogni giorno sono attivi N sportelli per la ricezione delle pratiche. L'accesso agli sportelli viene consentito solo se esiste uno sportello libero (nessun utente allo sportello). Per ottenere l'accesso, l'utente deve dichiarare all'ingresso quante pratiche vuole depositare (al massimo 10) e se c'e' uno sportello libero, viene indirizzato verso questo. In ciascun momento deve essere possibile conoscere il numero di pratiche evase da ciascuno sportello.
- Si modellino in particolare le seguenti operazioni:
  - Presentazione dell'utente all'ingresso: forniti Nome e Cognome e Numero delle pratiche da depositare, il sistema ricerca uno sportello libero e, se esiste, assegna all'utente il numero dello sportello e lo lascia entrare.
  - Presentazione dell'utente allo sportello: quando l'utente si presenta allo sportello, il sistema deve aggiornare il numero delle pratiche evase dallo sportello aggiungendo il numero di pratiche presentate dall'utente.
  - Uscita dell'utente dallo sportello: quando un utente a cui era stato assegnato uno sportello esce, il sistema deve aggiornare lo stato degli utenti presenti allo sportello.

Tipi, variabili

- Tipi definiti dall'utente
  [Persone,Sportelli,Pratiche]
  - Sportelli=1..N
  - Pratiche=1..10
  - Persone={(Nome,Cognome)|Nome e Cognome indicano una persona}
- Variabili che descrivono lo stato del sistema:
  - assegnazioni:funzione parziale che associa ad uno sportello la persona che lo sta occupando
  - evasioni: funzione totale che associa ad uno sportello il numero di pratiche evase

Stato, inizializzazione, successo

Tribunale

assegnazioni: Sportelli $\rightarrow$ Persone
evasioni: Sportelli $\rightarrow \mathbb{N}$

InitTribunale

Tribunale'

assegnazioni' = $\varnothing$
evasioni'={(n,0) | n $\in$ Sportelli}

Successo

rep!: Report

rep! = 'Okay'

Presentazione dell'utente all'ingresso

Precondizioni:
1) c'è uno sportello libero

PresentazioneIngresso

ΔTribunale
persona?: Persone
sportello!: Sportelli

sportello! ∉ dom assegnazioni
assegnazioni'=assegnazioni ∪ {sportello! ↦ persona?}
evasioni' = evasioni

NessunoSportelloLibero

ΞTribunale
rep!: Report

∄ sportello: Sportelli•
(sportello ∉ dom assegnazioni)
rep! = 'Nessuno sportello libero'

PresentazioneUtenteIngresso ≙ PresentazioneIngresso∧ Successo
∨
NessunoSportelloLibero

Presentazione dell'utente allo sportello

Precondizioni:
1) l'utente deve assere assegnato a uno sportello

PresentazioneSportello

ΔTribunale
persona?: Persone
pratichePersona?: Pratiche

assegnazioni'=assegnazioni
∀ sportello: Sportelli •
(sportello ∈ assegnazioni^{-1}(| persona |) ↦
evasioni' − evasioni ⊕ {sportello ↦ evasioni(sportello)+pratichePersona})

NessunoSportello

ΞTribunale
persona?: Persone
rep?: Report

|assegnazioni^{-1}(| persona |)|=0
rep?='L'utente non è assegnato ad aluno sportello'

PresentazioneUtenteSportello ≙ PresentazioneSportello∧ Successo
∨
NessunoSportello

Uscita dell'utente

Precondizioni:
1) l'utente deve assere assegnato ad uno sportello

UscitaSportello

ΔTribunale
persona?: Persone

|assegnazioni ^{1}(| persona |)|=1
evasioni'=evasioni
∀ sportello: Sportelli •
(sportello ∈ assegnazioni^{-1}(| persona |) ↦
assegnazioni' = assegnazioni \ {sportello ↦ persona?}

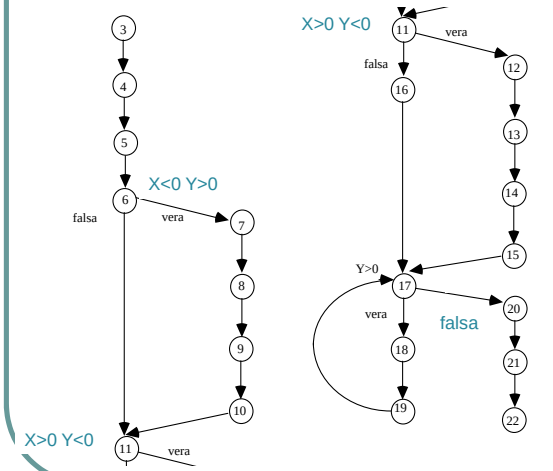
UscitaUtenteSportello ≙ UscitaSportello∧ Successo
∨
NessunoSportello

Esercizio 11

- Si individui un'insieme di casi di test di dimensione minima secondo il criterio di copertura dei comandi per il programma a fianco.
- Dire se e come andrebbe variato tale insieme di dati di test per soddisfare il criterio di copertura delle decisioni.

```
1.  program main;
2.  var X, Y, ABS_P: integer;
3.  begin
4.        read(X);
5.        read(Y);
6.        if X < 0 and Y >0
7.        then
8.              begin     X := -X;
9.                        ABS_P:=1
10.             end;
11.       if Y < 0 and X >0
12.       then
13.             begin     ABS_P:=1;
14.                       Y := -Y
15.             end
16.       else ABS_P:=1;
17.       while Y>0 do
18.             begin     ABS_P:=ABS_P+X;
19.                       Y := Y-1
20.             end;
21.       writeln(ABS_P);
22. end.
```

Soluzione



- Per coprire tutti i nodi e' sufficiente eseguire il programma con dati di test che soddisfano le condizioni:
  - **Caso 1:** X<0 Y>0
  - **Caso 2:** X>0 Y<0
- Il test sintetizzato può essere: T::={(X=-1;Y=2), (X=2; Y=-1)}
- I due casi coprono anche tutti gli archi.

Esercizio 12

- Dato il programma a fianco (in Pascal), si determini un insieme (minimo) di cammini da coprire per soddisfare il criterio di copertura di tutti gli usi.

```
1  program undici;
2  var   A,B,C: Integer;
3        X,Y: real;
4  begin
5        Read(A);
6        Read(B);
7        if (B-A*C)>0
8              then   begin
9                     X:=-B+sqrt(B-A*C);
10                    Y:=-B-sqrt(B-A*C);
11             end
12       else X:=-Y;
13       while (A-B > 0) do
14             begin
15             A:= A-C;
16             X:=X-1;
17             Y:=Y-2;
18             end
19 end.
```


Soluzione

		def	use	du(A)	du(B)	du(C)	du(X)	du(Y)
1	**program** undici;							
2	**var** A,B,C: Integer;							
3	X,Y: real;							
4	**begin**							
5	Read(A);	A		7,9,10, 13,15				
6	Read(B);	B			7,9,10,1 3			
7	**if** (B-A*C)>0		A,B,C					
8	**then begin**							
9	X:=-B+sqrt(B-A*C);	X	A,B,C				16	
10	Y:=-B-sqrt(B-A*C);	Y	A,B,C					17
11	**end**							
12	**else** X:=-Y;	X	Y				16	
13	**while** A-B > 0		A,B					
14	**do begin**							
15	A:= A-C;	A	A,C	13, 15				
16	X:=X-1;	X	X				16	
17	Y:=Y-2;	Y	Y					17
18	**end**							
19	**end**.							

Soluzione

- I cammini da coprire sono quindi:
  - 5-6-7-8-9-10-11-13-14-15 e 15-16-17-18-13-14-15 per la var. A
  - 6-7-8-9-10-11-13 per la var. B
  - 9-10-11-13-14-15-16, 12-13-14-15-16 e 16-17-18-13-14-15-16 per la var. X
  - 10-11-13-14-15-16-17 e 17-18-13-14-15-16-17 per la var. Y

Esercizio 13

- Si determinino le espressioni regolari D-U per ciascuna variabile del programma a fianco. Cosa suggerisce tale risultato?

```
program ventisette;
var     A,B,C: Integer;
        X,Y: real;
begin
        Read(A);
        Read(B);
        if (B-A*C)>0
                then    begin
                        X:=-B+sqrt(B-A*C);
                        Y:=-B-sqrt(B-A*C);
                        end
        else X:=-Y;
        while A-B > 0
        do      begin
                A:= A-C;
                X:=X-1;
                Y:=Y-2;
                end
end.
```


Soluzione

program ventisette;	A	B	C	X	Y
var A,B,C: Integer;	a	a	a		
X,Y: real;				a	a
begin					
Read(A);	d				
Read(B);		d			
if (B-A*C)>0	u	u	u		
then begin					
X:=-B+sqrt(B-A*C);	u	u	u	d	
Y:=-B-sqrt(B-A*C);	u	u	u		d
end					
else X:=-Y;				d	u
while A-B > 0	u	u			
do begin					
A:= A-C;	ud		u		
X:=X-1;				ud	
Y:=Y-2;					ud
end					
end.					

Espressioni regolari

- Per **A**:
  - adu(uu+ε) u (udu)*
- Per **B**:
  - adu(uu+ε) u (u)*
- Per **C**:
  - au(uu+ε) (u)*
- Per **X**:
  - a(d+d)(ud)*
- Per **Y**:
  - a(d+u)(ud)*
- Osservazioni:
  - Ci sono usi di C senza che questa variabile sia stata definita.
  - Ci possono essere usi di Y senza che questa variabile sia stata definita.