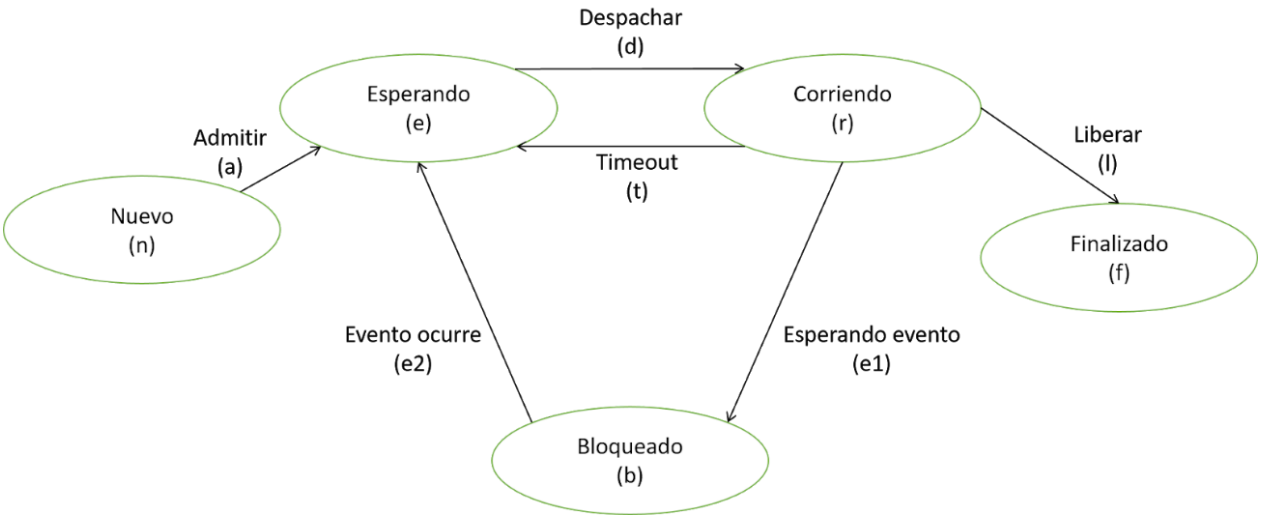
Ejercicios de Laboratorio (2)
Programación Lógica con Prolog

Universidad de Santiago de Chile
Departamento de Ingeniería Informática
Paradigmas de Programación (2021-2)
Prof. Daniel Gacitúa Vásquez

Ejercicio 1

Los programas en ejecución en los sistemas operativos Unix usan el modelo de procesos de 5 estados. Todo proceso inicia en el estado *"Nuevo (n)"* y puede cambiar de estados según lo indicado en el grafo de la figura. Por ejemplo, para que un proceso pase del estado *"Esperando (e)"* a *"Corriendo (r)"*, debe ser *"despachado (d)"* por el planificador de tareas.



Se está probando un nuevo planificador de tareas experimental para Unix, y para verificar su correcto funcionamiento se cuenta con los siguientes logs de salida:

PID	Estado Actual	Acciones del planificador
1001	n	[a,d,e1,e2,d,t]
1002	e	[d,t,e2,e1,l]
1003	b	[e2,d,t,l]

Para verificar el planificador, se le solicita que realice un programa en pseudo-Prolog que contenga lo siguiente:

1. Registrar las relaciones del modelo de 5 estados como hechos de Prolog

2. Crear un predicado que permita determinar si las acciones del nuevo planificador son válidas de acuerdo al modelo de 5 estados, tomando como parámetros de entrada el estado actual del proceso y una lista con las acciones del planificador**.** Ejemplos:

   `verificarLog(n, [a, d, l]). => true`

   `verificarLog(r, [t, e2, e1, d]). => false`

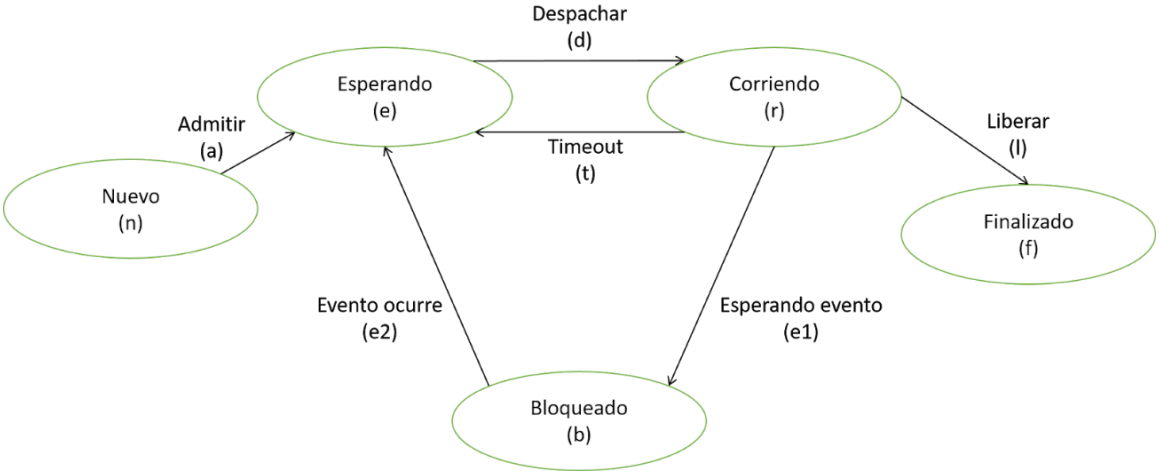
3. En su programa, indique claramente predicados, dominio, metas y cláusulas de Horn

4. Realizar la traza de los logs con PID 1001 y 1002 (indicados en la tabla) con su predicado `verificarLog`, indicando el resultado de salida

Grupo 1
Integrantes:
Daniel Catalan
Francisco Cea
Carlos Vera
Leo Vergara
Jonathan Soto

Desarrollo

1.-



%hechos

```
% Nodos
nuevo(n). o estado(n). o nodo(n).
esperando(e). o estado(e). o nodo(e).
corriendo(r). o estado(r). o nodo(r).
finalizado(f). o estado(f). o nodo(f).
bloqueado(b). o estado(b). o nodo(b).
```

%transiciones
admitir(a). o transicion(a).
despachar(d). o transicion(d).
liberar(l). o transicion(l).
timeout(t) o transicion(t).
esperandoEvento(e1). o transicion(e1).
eventoOcurre(e2). o transicion(e2).
definiciones de hechos redundantes

% aristas (transiciones)
% atomo1: estado actual.
% atomo2: estado siguiente.
% atomo3: transición.

```
arista(n, e, a).
arista(e, r, d).
arista(r, f, l).
arista(r, e, t).
arista(r, b, e1).
arista(b, e, e2).
```

2.- Crear un predicado que permita determinar si las acciones del nuevo planificador son válidas de acuerdo al modelo de 5 estados, tomando como parámetros de entrada el estado actual del proceso y una lista con las acciones del planificador. Ejemplos:

```
verificarLog(n, [a, d, l]). => true

verificarLog(r, [t, e2, e1, d]). => false
```

PID	Estado Actual	Acciones del planificador
1001	n	[a,d,e1,e2,d,t]
1002	e	[d,t,e2,e1,l]
1003	b	[e2,d,t,l]

verificarPlanificador(n, [a,d,e1,e2,d,t])

verificarPlanificador(e, [d,t,e2,e1,l])

verificarPlanificador(b, [e2,d,t,l])

daniel

/*verificar que en el estado actual se pueda realizar la transición de la cabeza de la lista

Si se puede hacer, se cambia el estado y se elimina la cabeza de la lista

caso contrario se retorna false

si la lista queda vacía, retornar true

*/

verificarPlanificardor(Nodo1, [X|Y]):-

arista(Nodo1, Nodo2, X) %revisa el estado actual, el siguiente y la cabeza de la lista

Nodo1 is Y

,verificarPlanificador(Nodo1, [X | Y]), !.

verificarPlanificador(N, [X|Y]):- arista(N, N1, [X]).
verificarPlanificador(N, [X|Y]):- arista(N, N1, [X]), append(Y, [X], Y1),
verificarPlanificador(N1, [X, Y1]).

```
% solucion profe

verificarLog(_, []).

verificarLog(E1, [H|T]):-

      arista(E1, E2, H), verificarLog(E2, T).
```