Universidad Nacional de Córdoba

FACULTAD DE CIENCIAS EXACTAS FÍSICAS Y NATURALES

CARRERA INGENIERÍA EN COMPUTACIÓN

TITULO A DEFINIR

Práctica Profesional Supervisada

Alumnos:
Casas, Nicolás
40574806, nicolas.casas@mi.unc.edu.ar, 3584305868
Ciarrapico, Nicolás Valentin
39880567, nicolas.ciarrapico@mi.unc.edu.ar, 2994840005

Supervisor: Dr. Ing. Orlando Micolini
Tutor: Ing. Luis Ventre

26 de octubre de 2022

Índice

1. Introducción

2. Desarrollo

2.1. Redes de Petri

Las Redes de Petri son modelos matemáticos utilizados para la representación de sistemas con paralelismo, concurrencia, sincronización e intercambio de recursos[5]. La red de Petri esencial fue definida por Carl Adam Petri. Son una generalización de la teoría de autómatas que permite expresar un sistema como eventos concurrentes.

Las redes de Petri están fuertemente asociadas a la teoría de grafos, ya que las mismas pueden representarse como un grafo dirigido bipartito compuesto por cuatro elementos:

- *Plazas*: Representan estados del sistema. El estado de una plaza está dado por la cantidad de marcas o tokens que esta contiene.
- *Token*: Figuran como puntos negros dentro de las plazas. Estos representan el valor específico de una condición o estado y generalmente se traducen en la presencia o ausencia de algún recurso del sistema.
- *Transiciones*: Representan el conjunto de sucesos cuya ocurrencia produce la modificación de los estados de las plazas, y en consecuencia del estado global del sistema.
- *Arcos*: Indican las interconexiones entre las plazas y las transiciones, estableciendo el flujo de tokens que sigue el sentido de la flecha.

2.1.1. Red de Petri Generalizada

Una Red de Petri generalizada no marcada es una cuádrupla $< P, T, Pre, Post >$ donde:

- $P = P_1, P_2, \ldots, P_n$ es un conjunto finito, no vacío, de plazas.
- $T = T_1, T_2, \ldots, T_m$ es un conjunto finito, no vacío, de transiciones, donde $P \cap T = \emptyset$, i.e. los conjuntos P y T son inconexos.
- $Pre : PxT \rightarrow \mathbb{N}^P$ es la función de incidencia de entrada y
- $Post : PxT \rightarrow \mathbb{N}^P$ es la función de incidencia de salida.

$Pre(p_i, t_j)$ contiene el peso del arco que va de P_i a T_j.
$Post(p_i, t_j)$ contiene el peso del arco que va de T_j a P_i.

2.1.2. Red de Petri Marcada

Una Red de Petri Marcada está definida por el par (N, M), donde N es una Red de Petri y $M : P \rightarrow \mathbb{N}^P$ (donde $|P| = p$ es una aplicación llamada **marcado**. $m(N)$ define el marcado de la RdP y m_{pi} indica el marcado de la plaza p_i, es decir, el número de tokens contenido en la plaza i. La marca inicial se denota m_0 y da la cantidad inicial de tokens en todas las plazas de la red, por lo que especifica el estado inicial del sistema.

2.1.3. Tipos de Arcos

- *Común*: Consume o produce una cierta cantidad de tokens de una plaza, de acuerdo al peso del mismo.
- *Inhibidor*: Siempre en la dirección plaza a transición. En el caso que el marcado de la plaza sea mayor o igual al peso del arco, la transición no está sensibilizada. No consume tokens al producirse el disparo de la transición asociada.

- *Lector*: Siempre en la dirección plaza a transición. La transición está sensibilizada si la marca en la plaza es mayor al peso del arco. No consume tokens al producirse el disparo de la transición asociada.
- *Reset*: Siempre en la dirección plaza a transición. Consume todos los tokens de la plaza al dispararse la transición.

2.1.4. Matrices

Para una red con n plazas y m transiciones, mas matrices tienen un tamaño nxm. Cada fila representa una plaza, mientras que cada columna representa una transición. Se conforman de la siguiente manera:

- *Matriz de Incidencia*: Está compuesta por las matrices I^+ e I^-, las cuales son función de los arcos comunes.

  En la matriz I^+, denominada matriz de incidencia de entrada o *post*, cada elemento $post(P_i, T_j)$ contiene el peso del arco que va desde T_j a P_i. Indica la cantidad de tokens generados al disparar la transición.

  En la matriz I^-, denominada matriz de incidencia de salida o *pre*, cada elemento $pre(P_i, T_j)$ contiene el peso del arco que va desde P_i a T_j. Indica la cantidad de tokens consumidos por la transición al realizar el disparo.

  Finalmente, la matriz de incidencia I se forma de la siguiente forma:

  $$I = I^+ - I^-$$

- *Matriz de inhibición*: Contiene en cada uno de sus elementos $inh(P_i, T_j)$, el peso del arco de inhibición que va desde $P_i a T_j$
- *Matriz de Reset*: Contiene en cada uno de sus elementos $res(P_i, T_j)$ un 1 si existe un arco de reset que va desde P_i a T_j
- *Matriz de lectura*: Contiene en cada elemento $lec(P_i, T_j)$ el peso del arco lector que va desde P_i a T_j.

2.1.5. Sensibilizado de una transición

Una transición está sensibilizada si todas las plazas de entrada a la transición tienen una marca igual o mayor al peso del arco que une cada plaza con la transición.

Previo a expresar la condición de sensibilizado de manera general, son necesarias las siguientes definiciones:

- $\bullet T_j$ es el conjunto compuesto por las plazas entrante a T_j
- $T_j \bullet$ es el conjunto compuesto por las plazas salientes a T_j.
- $M_k(P_i)$ es el marcado de la plaza P_i antes de disparar la transición T_j.
- $M_{k+1}(P_i)$ es el marcado de la plaza P_i después de dispara la transición T_j.
- w_{ij} es el peso del arco $P_i \rightarrow T_j$
- w_{ji} es el peso del arco $T_j \rightarrow P_i$

De esta forma, el sensibilizado de una transición T_j se expresa como:

$$T_j \text{ está sensibilizada } sii\ \forall P_i \in \bullet T_j \rightarrow M_k(P_i) \geq w_{ij}$$

Lo definido anteriormente se cumple para redes cuyos arcos son todos comunes. Para redes con arcos inhibidores y/o de lectura cambia la condición de sensibilizado. De esta forma, dependiendo del tipo de arco deben cumplirse ciertas condiciones:

- *Arco de inhibición*: El marcado de la plaza de la cual parte debe es menor que el peso del arco.
- *Arco de lectura*: El marcado de la plaza de la cual parte debe ser mayor o igual al peso del arco.

Por su parte los arcos de reset no alteran la condición de sensibilizado de una transición.

2.1.6. Disparo de una transición

Dada una marca $M_k(P)$, cualquier transición que se encuentre sensibilizada puede ser disparada, este disparo nos llevará a una nueva marca $M_{k+1}(P)$ dada por:

$$M_{k+1}(P) = M_k(P) + I^+(P_i, T_j) - I^-(P_i, T_j) \forall P_i \in P$$

Al disparar la transición T_j se extraen tantos tokens de $\bullet T_j$ como indiquen los arcos que unen estas plazas con T_j. Se añaden a $T_j \bullet$ la cantidad de tokens que indiquen los arcos que unen a T_j con estas plazas. El disparo de una transición T_j se denota $M_k \to T_j \to M_{k+1}$.

2.1.7. Ecuación de estado

La ecuación de estado representa matemáticamente el comportamiento dinámico del sistema [4]. Esta permite obtener el estado del sistema luego del disparo de una transición. Se utiliza el marcado en un instante k para calcular el marcado de la red en un instante de tiempo $k + 1$. Por lo tanto la ecuación de estado para una Red de Petri con n plazas y m transiciones se define de la siguiente forma:

$$M_{k+1} = M_k + I * \sigma$$

Siendo:

1. I la matriz de incidencia.
2. σ vector de disparo. Tiene dimensión m x 1 y contiene 1 en la posición de la transición que se quiere disparar.
3. M_k: vector de marcado actual.
4. M_{k+1}: vector de marcado del estado siguiente.

2.1.8. Extensión de la ecuación de estado

La ecuación de estado descrita previamente es útil únicamente en presencia de arcos comunes. Para representar matemáticamente la existencia de los demás arcos nombrados anteriormente se requiere de una matriz para indicar la conexión plaza transición para cada uno de estos.

- *Vector de transiciones des-sensibilizadas por arco inhibidor*: Es un vector de valores binarios de dimensión $m \times 1$, que indica con un cero cuáles transiciones están inhibidas por el arco y con un uno cuales no.
- *Vector de transiciones des-sensibilizadas por arco lector*: Es un vector de valores binarios de dimensión m x 1, que indica con un cero cuáles transiciones están inhibidas por el arco con un uno las que no.
- *Vector transiciones des-sensibilizadas por tiempo*: Es un vector de valores binarios de dimensión m x 1, que indica con un cero cuáles transiciones están inhibidas porque no se ha alcanzado o se ha superado el intervalo de tiempo transcurrido desde que la transición fue sensibilizada.

- *Vector de transiciones reset*: Es un vector de valores enteros de dimensión m x 1, que tiene el valor de la marca de la plaza que se quiere poner a cero, mientras que las otras componentes son uno.

Finalmente utilizando los vectores descriptos anteriormente se puede obtener el vector de sensibilizado extendido E_x:

$$E_x = E \text{ } and \text{ } B \text{ } and \text{ } L \text{ } and \text{ } Z$$

Para poder introducir el brazo reset hay que multiplicar elemento a elemento (#) al vector que resulte de la conjunción por A. Obteniendo así la ecuación de estado extendida:

$$M_{k+1} = M_k + I * ((\sigma \text{ } and \text{ } E_x)\#A)$$

2.1.9. Propiedades de las Redes de Petri

2.1.9.1 Propiedad de limitación

Dada una Red de Petri PN, se dice que una plaza P_i está **k-limitada** por una marcado inicial M_0 si hay un entero natural k tal que, para todas las marcas alcanzables desde M_0, el número de tokens en P_i no es mayor que k. Es decir:

$$\exists \ k \in \mathbb{N} \ / \ \forall M \in marcados(PN) \rightarrow M(P) \leq k$$

Una Red de Petri PN está limitada para un marcado inicial M_0 si todos los lugares están limitados para M_0. Es decir, PN está limitada por k si todas sus plazas están limitadas por k.

A partir de esta definición surgen varios conceptos, entre los cuales se encuentran los siguientes:

- Una Red de Petri es **segura** si todas sus plazas son **1-limitadas**
- Una Red de Petri es **cíclica** si siempre existe la posibilidad de alcanzar el marcado inicial desde cualquier otro marcado alcanzable.
- Una Red de Petri es **repetitiva** si existe una secuencia de disparos que contiene a todas las transiciones y que lleva a la red desde el marcado actual al mismo marcado.
- Una Red de Petri es **conservativa** si se cumple que el número de tokens en el marcado es siempre el mismo.

2.1.9.2 Propiedad de vivacidad

La **vivacidad** de una transición indica que, en todo instante de la evolución de la red, su disparo es posible. Este concepto es particularmente relevante ya que determina si la ejecución de la red puede o no detenerse en un estado determinado. A partir de esto se puede definir la vivacidad de una red de Petri. Esta propiedad indica que una red es viva para un marcado si todas sus transiciones lo son.

Por otro lado, la **cuasi-vivacidad** de una transición expresa la posibilidad de dispararla al menos una vez a partir de un marcado inicial M_0. De la misma manera que para el caso de la vivacidad, una red de Petri es cuasi-viva si todas sus transiciones lo son.

Gracias a esta última definición, se puede definir la vivacidad en función de la cuasivivacidad de la siguiente manera: una transición es viva si la misma es cuasi-viva en la red para todo marcado alcanzable desde M_0.

La vivacidad está directamente asociada con la ausencia de **deadlock** o **interbloqueo**. En términos generales, el deadlock es el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema o bien se comunican entre ellos. En el caso de una red de Petri, esto suele ocurrir cuando dos o más transiciones esperan mutuamente por el disparo de la otra, produciendo el bloqueo permanente de esa porción de la red. Una red de Petri viva garantiza la ausencia de interbloqueo sin importar la secuencia de disparos.

2.1.9.3 Propiedad de alcanzabilidad

La **alcanzabilidad** de una Red de Petri es fundamental para el análisis de las propiedades dinámicas de un sistema. A grandes rasgos, permite determinar si el sistema modelado puede alcanzar un determinado estado.

Un marcado M_i es alcanzable desde el marcado inicial M_0 si existe una secuencia finita de disparos que me haga llegar a esa marca. Un marcado M_i es alcanzable desde M_0 si existe una secuencia finita de disparos σ tal que $M_0 \xrightarrow{\sigma} M_i$.

El conjunto de marcados alcanzables puede ser representado mediante un grafo, en el cual cada nodo representa un marcado. El arco entre estos nodos representa la transición que se necesita disparar para llegar de un marcado a otro.

Ejemplo:

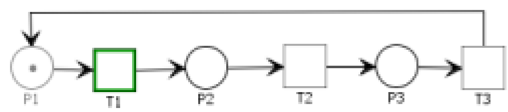

Figura 1: Red de Petri de tres estados.

Esta red cuenta con tres estados:

1. Marcado inicial o al disparar T3 desde m_2: $m_0 = [1, 0, 0]$
2. Marcado al disparar T1 desde m_0: $m_1 = [0, 1, 0]$
3. Marcado al disparar T2 desde m_1: $m_2 = [0, 0, 1]$

Con estos estados es posible generar el siguiente grafo:

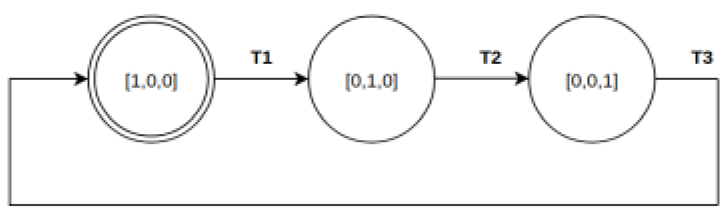

Figura 2: Grafo de alcanzabilidad.

2.1.9.4 Cobertura de una Red de Petri

Si la red de Petri no es acotada habrá plazas cuya cantidad de tokens aumentará indefinidamente, lo que también se repetirá en los nodos del grafo de alcanzabilidad, por lo que el algoritmo para obtener el árbol de alcanzabilidad no convergerá. Para estos casos existe otro tipo de análisis denominado **grafo de cobertura**.

El grafo de cobertura consiste en graficar todas las marcas posibles de la red, pero colapsando en un único marcado genérico, representado con el símbolo ω, aquellas plazas cuya cantidad de tokens crecerá infinitamente.

2.1.10. Invariantes

Las invariantes de una red son propiedades independientes tanto del marcado inicial como de la secuencia de disparos, y pueden asociarse a ciertos subconjuntos de plazas o de transiciones. De esta forma surgen dos conceptos:

- **Invariante de plaza** o **p-invariante**: Se llama así a un conjunto de plazas si dado un determinado vector de ponderación con enteros positivos o cero se cumple que:

$$q_1 \text{ x } M(P_1) + q_2 \text{ x } M(P_2) + \cdots + q_n \text{ x } M(P_n) = K$$

  Siendo K constante para todo marcado. Es un componente conservativo independiente del marcado inicial. Sin embargo, el valor de la constante sí depende del marcado inicial.

- **Invariante de transición** o **t-invariante**: conjunto de transiciones que una vez disparadas secuencialmente generan el mismo marcado del cual habían partido, pudiendo disparar la secuencia indefinidamente.

2.2. Reinforcement Learning

Reinforcement Learning (RL) [6] es una rama del *machine learning* donde el aprendizaje se da interactuando con el ambiente. Es un aprendizaje orientado a objetivos, donde al algoritmo no se le enseña qué acciones debe tomar; sino que aprende de las consecuencias de sus acciones.

Este tipo de aprendizaje se complementa con los ya estudiados anteriormente como se ve en la siguiente Fig.3:

Figura 3: Tipos de aprendizajes.

El Reinforcement Learning entonces, intentará hacer aprender a la máquina basándose en un esquema de "premios y castigos" en un entorno en donde hay que tomar acciones y que está afectado por múltiples variables que cambian con el tiempo [2]

2.2.1. Componentes del RL

- *El agente:* Es el modelo que queremos entrenar para que aprenda a tomar decisiones
- *Ambiente:* Es el entorno en donde interactúa y "se mueve" el agente. El ambiente contiene las limitaciones y reglas posibles en cada momento.
- *Acción:* Son las posibles acciones que puede tomar en un momento determinado el Agente. (Cambiar de estado).

- *Estado:* Es el indicador de cómo se encuentran los diversos elementos que componen el ambiente en un momento dado.
- *Recompensas:* Cada acción tomada por el Agente tendrá como consecuencia un refuerzo positivo o negativo que orientará al Agente hacia la forma correcta de comportarse.
- *Política:* Define el comportamiento del agente en el ambiente.

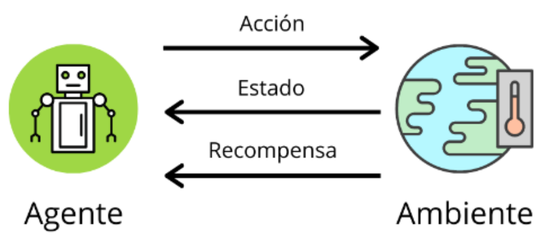

Figura 4: Interacción entre el agente y el entorno.

El agente se encuentra en un primer momento en un "estado inicial" y luego realiza una acción, lo cual genera que esto influya en el ambiente, luego de esto el agente obtiene dos cosas: Un nuevo estado y la recompensa. Si esta ultima es negativa el agente actuara de forma distinta frente a dicha situación y si es positiva reforzara este comportamiento. Esta interacción se ve mas claramente en la Fig.4.

Al finalizar la instancia de aprendizaje todo el conocimiento aprendido es almacenado en la polıtica.

Figura 5: Interacción entre el agente y el entorno.

Se debe lograr un equilibrio entre la recompensa y el agente, dado que si el agente recibe una recompensa lo suficientemente alta el mismo siempre realizará la misma acción no consiguiendo generalizar el sistema. Fig.5.

2.2.2. Tipos de algoritmos de RL

Existen dentro del aprendizaje por refuerzo ciertos algoritmos a emplear, algunos de ellos son:

2.2.2.1 Q-Learning(Value Learning):

El objetivo principal al entrenar el modelo a través de las simulaciones es ir completando una matriz de Políticas de manera que las decisiones que tome nuestro agente obtengan "la mayor recompensa" evitando el sobreajuste.

- A la política se la denomina Q
- Q(estado, acción) nos indica el valor de la política para un estado y una acción determinados.

Para completar la matriz de políticas se utiliza la ecuación de Bellman Fig.6.

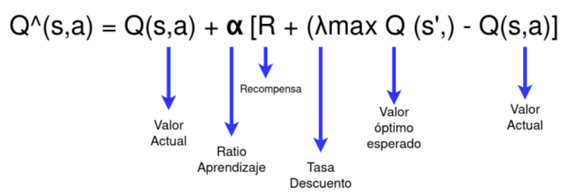

Figura 6: Ecuación de Bellman.

Esta ecuación determina cómo se irán actualizando las políticas Qˆ(s,a), en base a su valor actual más una recompensa recibida como consecuencia de dicha acción. Hay dos ratios que afectan a la manera en que influye esa recompensa: el ratio de aprendizaje, que regula "la velocidad" en la que se aprende, y la "tasa de descuento" que tendrá en cuenta la recompensa a corto o largo plazo.

2.2.2.2 Policy learning

Este método de aprendizaje por refuerzo tiene como principal diferencia a Value Learning (Q-Learning) que en esta última se busca tener una Red Neuronal/Política que aprenda a aproximar la función Q [1], para obtener un valor $Q_(s,a)$ de un estado dada una acción, y luego usamos este valor para inferir cuál es la mejor acción a tomar, esta es nuestra política.

Por otra parte **Policy Learning** busca directamente aprender la política pudiendo usar una NN o, en casos más simples, una matriz para luego poder alimentar la misma con una entrada y como salida se tendrá qué acción debemos tomar. Esto simplifica la situación, ya que para obtener la acción a tomar, es decir la acción que maximizar´a la recompensa dado un estado, simplemente se debe muestrear desde la función de política sin necesidad de realizar numerosas valuaciones.

En resumen en **Value Learning** se aproxima una función Q y se la usa para inferir la política óptima, mientras que en **Policy Learning** se optimiza la política de forma directa.

La política en esta metodología será alimentada con el estado actual, y como salida tendrá las probabilidades correspondientes a cada acción posible.

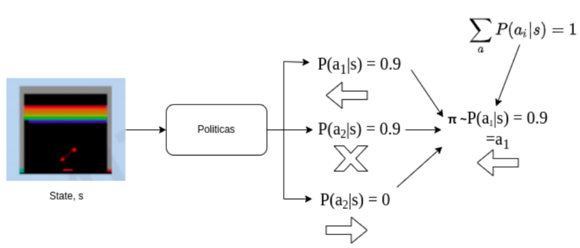

Figura 7: Policy gradient.

En la figura 7 se tiene como ejemplo las probabilidades de los distintos tipos de acciones que puede tomar el agente (la barra) con el fin de pegarle a la bola roja. Aquí la opción óptima es la de moverse a la izquierda, pero dado que esta es una distribución de probabilidad, la vez que se muestree podría llegar a elegirse quedarse en el lugar (acción 2) dado que también cuenta con un valor no nulo de probabilidad.

Este método de RL puede trabajar con valores continuos. Siguiendo con el ejemplo previo, podría no solo obtenerse la dirección a la cual moverse, sino también la velocidad con la cual hacerlo. Esto puede realizarse visualizando la salida como una distribución probabilística dependiendo de cual se adapte más a la situación. En el ejemplo siguiente se toma una distribución gaussiana.

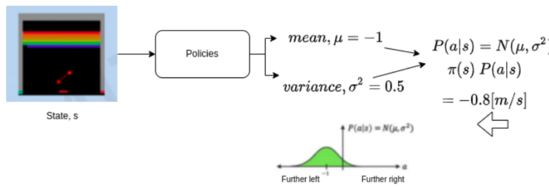

Figura 8: Policy gradient.

En la Fig.8 puede observarse que el agente debe moverse hacia la izquierda con una velocidad cercana a -1 m/s. Al muestrear en esta distribución, se puede notar que la velocidad en concreto que tomará el agente es de 0.8 m/s hacia la izquierda. De esta forma también se puede ver que a pesar de que la media de esta distribución es -1, no estamos limitados a ese número exacto.

Todo esto abre la posibilidad para escenarios donde puede llegar a tenerse un sin fin de acciones posibles a tomar.

2.3. Redes de Bayes

2.3.1. Teorema de Bayes

Sea $\{A_1, A_2, \ldots, A_i, \ldots, A_n\}$ un conjunto de sucesos mutuamente excluyentes y exhaustivos tales que la probabilidad de cada uno de ellos es distinta de cero ($P[A_i] \neq 0$ para $i = 1, 2, \ldots, n$). Si B es n suceso cualquiera del que se conocen las probabilidades condicionales $P(B|A_i)$ entonces la probabilidad $P(A_i|B)$ viene dada por la expresión:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

donde:

- $P(A_i)$ son las probabilidades a priori.
- $P(B|A_i)$ son las probabilidades de B en la hipótesis A_i.
- $P(A_i|B)$ son las probabilidades a posteriori.

2.3.2. Fórmula de Bayes

Con base en la definición de probabilidad condicionada se obtiene la Fórmula de Bayes, también conocida como Regla de Bayes:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{k=1}^{n} P(B|A_k)P(A_k)} \cdots$$

Esta fórmula nos permite calcular la probabilidad condicional $P(A_i|B)$ de cualquiera de los eventos A_i dado B.

2.3.3. Redes Bayesianas

Es un grafo que representa la relación entre las variables de un problema, permitiendo una representación compacta y son de ayuda en la toma de decisiones [3]. Están conformadas por:

- *Nodos:* círculos que representan una variable aleatoria.
- *Flechas:* relaciones causales entre las variables aleatorias (conexiones entre nodos).

Las redes Bayesianas nos permiten realizar inferencias.

Referencias

[1] Alexander Amini. Deep reinforcement learning. mit, 6:s191, 2019.

[2] Juan Ignacio Bagnato. Aprendizaje por refuerzo. http://aprendemachinelearning.com/aprendizaje-por-refuerzo/.

[3] José Luis Iglesias Feria. Ia probabilidad - redes bayesianas. https://www.youtube.com/playlist?list=PLYWD-VqrD5BCr-QeESSOvpEqkAcvQOrQO, 2018.

[4] Micolini Orlando, Cebollada Marcelo, Eschoyez Maximiliano, Ventre Luis O., and Schild Marcelo. Ecuación de estado generalizada para redes de petri no autónomas y con distintos tipos de arcos. *XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016)*, 2016.

[5] J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Independently Published, 2019.

[6] Simonini Thomas. Deep reinforcement learning course. https://simoninithomas.github.io/deep-rl-course/.