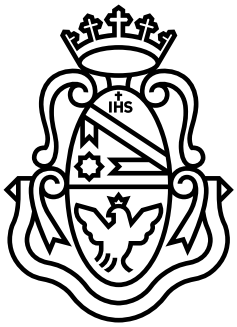
UNIVERSIDAD NACIONAL DE CÓRDOBA
Facultad de Matemática, Astronomía y Física

Análisis de Diagnosticabilidad en Sistemas Probabilísticos

Autor: Patricio Hubmann
Directora: Laura Brandán Briones



1 de julio de 2015

Resumen

Un aspecto fundamental en sistemas computacionales es poder diseñar mecanismos automáticos de detección de fallas. Debido al incremento en los requerimientos de confiabilidad (particularmente en sistemas críticos), muchos métodos han sido propuestos para realizar un análisis exhaustivo de las fallas. Dado un modelo de un sistema con observabilidad parcial, la propiedad de diagnosticabilidad garantiza que toda falla pueda ser inequívocamente detectada basado solamente en una serie de observaciones. Esto se hace a partir de un monitoreo sistemático del sistema (información de sensores o logs del sistema). Cabe destacar que nos referimos a fallas inherentes al sistema que no pueden ser excluidas en la etapa de diseño (ej. un corte de electricidad o un defecto de hardware). Se han realizados muchos intentos para determinar la verificación formal de diagnosticabilidad, pero la mayoría de los trabajos asumen sistemas sin probabilidades, lo cual motiva la presente propuesta. En este trabajo nos focalizamos en el análisis de diagnosticabilidad en sistemas probabilísticos. En primer lugar presentamos el problema para sistemas discretos, luego introducimos probabilidades en los sistemas y finalmente analizamos el caso de sistemas probabilistas no deterministas.

Índice

1. Introducción **9**

2. LTS **13**
2.1. Introducción a LTS 13
2.1.1. Trazas, caminos y lenguajes 14
2.1.2. Restricciones sobre los modelos 16
2.2. Diagnosticabilidad sobre LTS 16
2.2.1. Método "Twin-Plant" 17

3. pLTS **21**
3.1. Introducción a pLTS 21
3.1.1. Probabilidades de trazas y caminos en un pLTS 21
3.2. Diagnosticabilidad y Diagnosis en pLTS 22
3.2.1. Diagnosticador estocástico 23
3.2.2. Construcción del Diagnosticador estocástico 23
3.2.3. Diagnosticabilidad usando el Diagnosticador estocástico 26
3.2.4. Comportamiento asintótico de la cadena de Markov . . 29
3.2.5. Diagnosis usando el Diagnosticador estocástico 33
3.2.6. Análisis de los valores de Diagnosticabilidad calculados 34

4. MDP **37**
4.1. Introducción a MDP 37
4.1.1. Representación gráfica 38
4.1.2. Caminos, trazas y lenguajes en MDP 38
4.1.3. Estrategias 39
4.1.4. pLTS derivado de un MDP con una estrategia sin memoria 40
4.1.5. Probabilidad de caminos y trazas en un MDP 41
4.2. Diagnosticabilidad sobre MDP 43
4.2.1. Diagnosticador de decisión 43
4.2.2. Utilidad del Diagnosticador de decisión 44
4.2.3. Diagnosticabilidad usando el Diagnosticador de decisión 45
4.2.4. Extracción de la cadena de Markov 46
4.2.5. Comportamiento asintótico de la cadena de Markov . . 47
4.2.6. Problema de optimización 50
4.2.7. Observaciones finales sobre el algoritmo 53

5. Análisis de Complejidad **56**
5.1. Algoritmo presentado para LTS 56
5.2. Algoritmo presentado para pLTS 56
5.3. Algoritmo presentado para MDP 57

6. Ejemplo ilustrativo **60**
6.1. Presentación 60
6.2. Máquinas clasificadoras 61
6.3. Análisis de Diagnosticabilidad del modelo 63
6.3.1. Diagnosticador de decisión 64
6.3.2. Cadena de Markov y su análisis asintótico 65
6.3.3. Problema de optimización 69

7. Conclusiones **74**

Apéndices **77**

Apéndice Apéndices **77**

Apéndice A. Optimización con restricciones lineales **77**
A.1. Introducción 77
A.2. Identificación de vértices, aristas y caras 78
A.2.1. Vértices 78
A.2.2. Aristas y caras 79
A.3. Representación paramétrica 81
A.4. Cálculo de la solución óptima 82

1. Introducción

Los sistemas complejos a menudo exhiben fallas que son intrínsecas al sistema, es decir, fallas que no pueden ser excluidas desde la etapa de diseño. Existen dos áreas complementarias que nos permiten trabajar sobre este tipo de fallas. La primera es la diagnosis, la cual intenta bajo observaciones del sistema saber si una falla ocurrió o va a ocurrir. El proceso de diagnosis se realiza en forma paralela a la ejecución del sistema, es decir, se aplica en una etapa post entrega. La segunda es el análisis de la diagnosticabilidad, en el cual un modelo del sistema se analiza para intentar que toda falla quede unívocamente asociada a una serie de observaciones en particular, de tal manera de asegurar que el futuro proceso de diagnosis sea consistente.

Un modelo es diagnosticable si cumple con la propiedad de diagnosticabilidad, es decir, si la ocurrencia de cada falla puede ser determinada inequívocamente usando sólo información del comportamiento observable del mismo. Las observaciones del sistema están dadas por información de sensores, los cuales permiten al observador saber una parte de los eventos que ocurren en el sistema. En general, la utilización de estos sensores no es suficiente como para entender todo el comportamiento del sistema, ya sea por la imposibilidad de detectar algunos eventos o por el costo que pueda implicar su instalación. El uso de sensores divide los eventos de los sistemas en observables e inobservables. Las fallas dentro de los sistemas son eventos especiales, los cuales son modelados como presuntas transiciones que pueden llegar a ocurrir dentro del sistema, sin embargo no pueden ser detectadas mediante el uso de sensores. Ejemplos típicos incluyen sistemas complejos como plantas de energía, sistemas de control de aeronaves, equipamientos de desarrollo industrial, etc.

Claramente, la existencia de métodos automatizables para determinar la ocurrencia de estas fallas es importante en sistemas complejos. La propiedad de diagnosticabilidad ha sido estudiada por bastante tiempo y existen varios algoritmos para su verificación en sistemas discretos. El método inicial para la verificación de diagnosticablidad está dado en [16] y se basa en la construcción de un *Diagnosticador*, el cual es una máquina de estados con transiciones observables que permite estimar el estado actual del sistema y la posible ocurrencia de fallas, siguiendo sus trazas observables, lo cual permite realizar diagnosis del sistema aparte de la verificación de diagnosticabilidad. Más adelante, otros algoritmos fueron desarrollados, siendo el método "Twin-Plant" [12] uno de los más conocidos debido a su complejidad polinomial en el

número de estados del sistema. El método "Twin-Plant" consiste en la construcción de un *verificador* del sistema, el cual consiste en una composición en paralelo del sistema consigo mismo sincronizándose en eventos observables. Este verificador compara cada par de caminos con el mismo comportamiento observable en el sistema, buscando posibles trazas que evidencien la no diagnosticabilidad del sistema, es decir, una serie de observaciones que puedan ser de una traza fallida como también de otra traza sin fallas. Como contrapartida, con este método se pierde la posibilidad de realizar diagnosis del sistema.

Muchos trabajos se han enfocado en la verificación de diagnosticabilidad sobre sistemas discretos, ya sea buscando algoritmos más eficientes [20], tomando distintos formalismos para el modelado de los sistemas [13, 4], analizando el problema sobre sistemas distribuidos [17, 7, 4, 14], analizando el problema mediante *model checking* [6], etc. Sin embargo, el interés por sistemas que presentan información probabilista sobre el comportamiento del sistema es bastante reciente. La información probabilista resulta muy valiosa en términos de la diagnosticabilidad de un sistema, ya que nos permite distinguir cuáles trazas son más probables de ocurrir, sobre otras trazas cuya probabilidad de ocurrencia es muy baja. Un posible comportamiento del sistema que viola la propiedad de diagnosticabilidad alcanza para catalogar a todo un sistema como no diagnosticable, a pesar de que tal comportamiento sea quizás muy poco probable. En [19, 11] se definen nuevas nociones de diagnosticabilidad basadas en la probabilidad de ocurrencia de trazas no diagnosticables. En [15] se cuantifica el grado de diagnosticabiliad del sistema, basado en la probabilidad de ocurrencia de trazas no diagnosticables. La importancia práctica de este tipo de respuesta es que si el costo de incrementar la observabilidad [5] es demasiado alto, se posibilita tolerar sistemas con una baja probabilidad de trazas no diagnosticables.

Otro tipo de sistemas combinan elecciones no deterministas con probabilidades. Este tipo de sistemas también admite un análisis sobre su diagnosticabilidad, aunque el no determinismo de las acciones no permite dar el mismo tipo de respuestas sobre la diagnosticabilidad obtenidas sobre sistemas solamente probabilistas. El análisis de diagnosticabilidad sobre sistemas no deterministas con probabilidades necesita también de una estrategia que resuelva el no determinismo presente en el modelo. El único trabajo, de nuestro conocimiento, que hemos encontrado que trata con probabilidades y no determinsimo de eventos es [2], en donde se hace un análisis de diagnosticabilidad sobre sistemas con acciones controlables y se analiza la existencia de

al menos una posible estrategia que garantice la diagnosticabilidad del sistema. Por el contrario, el enfoque de diagnosticabilidad que presentamos en este trabajo consiste en derivar una estrategia que sirva de referencia como el peor caso que pueda llegar a darse con respecto a la elección de eventos en términos de diagnosticabilidad. Esta forma de encarar el problema se justifica al modelar comportamientos de un sistema que no pueden ser controlados, en los cuales se desea conocer el caso más desfavorable que puede darse para decidir si el sistema debe o no replantearse para obtener mejores resultados.

En este trabajo hacemos un resumen sobre los métodos utilizados para determinar si un sistema es o no diagnosticable y para cuantificar el nivel de diagnosticabilidad de sistemas completamente probabilistas. Luego, intentamos trasladar ese mismo análisis a sistemas probabilistas que añaden no determinismo.

La organización de este trabajo es la siguiente: en la Sección 2 analizamos diagnosticabilidad de sistemas representados como sistemas de transiciones etiquetados (LTS) para introducir los aspectos claves sobre diagnosticabilidad. Presentamos el método 'Twin-Plant" introducido en [12] para la verificación de diagnosticabilidad sobre LTS. En la Sección 3 comenzamos introduciendo probabilidades en los sistemas, modelándolos como sistemas de transiciones etiquetadas probabilísticos (pLTS). Sobre estos sistemas hacemos un análisis de diagnosticabilidad construyendo un Diagnosticador estocástico, y a partir de él extraemos una cadena de Markov para analizar su comportamiento asintótico, es decir cuando es sistema se estabiliza. El contenido de esta sección está basado en los trabajos [19, 11] y [15]. En la Sección 4 introducimos el aporte original de este trabajo, combinando probabilidades con no determinismo en las transiciones para analizar sistemas modelados como procesos de decisión de Markov (MDP). A partir del análisis de diagnosticabilidad presentado obtenemos una estrategia que resuelve el no determinismo del sistema de manera tal que el sistema probabilista resultante toma el comportamiento más desfavorable posible en términos de diagnosticabildad, lo cual permite decidir si es necesaria o no la reformulación del sistema. Conluyendo, en la Sección 5 hacemos un análisis sobre la complejidad de los algoritmos empleados y finalizamos con un ejemplo ilustrativo.

2. LTS

2.1. Introducción a LTS

Usamos para el modelado de sistemas los **Sistemas de Transiciones Etiquetadas** *(o LTS por sus iniciales en inglés)*. Partiendo desde un estado inicial, una secuencia de eventos permite evolucionar al LTS, y el conjunto de todas las secuencias de eventos permitidas por el sistema (llamado el *lenguaje* del sistema) capturan por completo su comportamiento. Formalmente, un LTS se define de la siguiente manera.

Definición (LTS)**.** *Definimos un Sistema de Transiciones Etiquetado o LTS como la 4-upla* $\mathcal{N} = \langle S, \Sigma, \delta, s_0 \rangle$ *donde:*

- S *es un conjunto finito de estados,*
- Σ *es un conjunto finito de eventos,*
- $\delta \subseteq S \times \Sigma \times S$ *es la relación de transición,*
- $s_0 \in S$ *es el estado inicial.*

Debido a la naturaleza de los problemas con los que tratamos, nos enfocamos en sistemas con observabilidad parcial, en los cuales el conjunto de eventos se encuentra particionado en dos partes: la primera compuesta por eventos *observables* y la segunda por eventos *inobservables*, denotadas con Σ_o y Σ_u respectivamente. También hay un grupo de eventos particulares que representan las fallas que puede tener un sistema, denotado con Σ_f, y asumimos que están incluidas dentro del conjunto de eventos inobservables ($\Sigma_f \subseteq \Sigma_u$), ya que en caso contrario su detección sería trivial (la falla es detectada en el momento en el que es observada). La Figura 1 muestra un esquema ilustrativo sobre la división del conjunto de eventos.

Una de las ventajas de utilizar LTS para modelar sistemas es su simple representación gráfica. Sea $\mathcal{N} = \langle S, \Sigma, \delta, s_0 \rangle$ un LTS, representamos gráficamente cada estado con un círculo, y cada transición con una flecha que une a dos círculos etiquetada con una acción. El estado inicial estará apuntado por una flecha sin origen. Para mayor comodidad, las flechas punteadas indicarán transiciones dadas por eventos inobservables, y las flechas zigzagueantes por fallas.

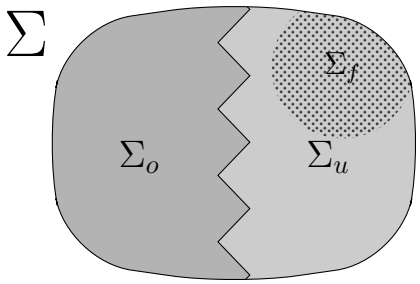


Figura 1: Conjunto de eventos

Ejemplo: La Figura 2 contiene la representación gráfica del LTS $\mathcal{N}$ dado por $\mathcal{N} = \langle\{0,1,2\},\{a,f,u\},\{(0,u,1),(0,f,2),(1,a,1),(2,a,2)\},0\rangle$, con $\Sigma_o = \{a\}$, $\Sigma_u = \{u,f\}$ y $\Sigma_f = \{f\}$.

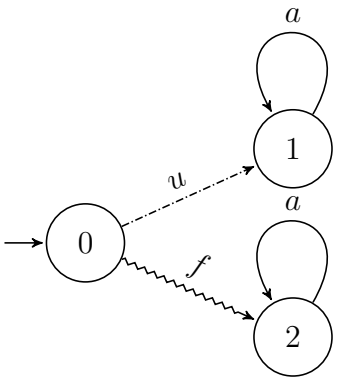


Figura 2

2.1.1. Trazas, caminos y lenguajes

Sea $\mathcal{N} = \langle S, \Sigma, \delta, s_0\rangle$ un LTS, entonces:

- con Σ^* y con Σ^ω denotamos al conjunto de todas las secuencias finitas e infinitas de elementos de Σ respectivamente, donde cada elemento de $\Sigma^* \cup \Sigma^\omega$ es llamado una *traza*, y la traza vacía se representa con la letra griega ϵ (epsilon)

- un *camino finito* sobre el LTS $\mathcal{N}$ es una secuencia $\rho = s_0{\cdot}a_1{\cdot}s_1{\cdot}\cdots a_n{\cdot}s_n$ finalizada siempre en algún estado, donde $s_i \in S$, $a_i \in \Sigma$ y $\forall i \in \{0,\ldots,n-$

$1\}, (s_i, a_{i+1}, s_{i+1}) \in \delta$, y con $Path^*_{\mathcal{N}}$ denotamos todos los caminos finitos sobre $\mathcal{N}$

- un *camino infinito* sobre el LTS $\mathcal{N}$ es una secuencia $\varrho = s_0 \cdot a_1 \cdot s_1 \cdots$ donde $s_i \in S$, $a_i \in \Sigma$ y $\forall i \in \mathbb{N}, (s_i, a_{i+1}, s_{i+1}) \in \delta$, y con $Path^\omega_{\mathcal{N}}$ denotamos todos los caminos infinitos sobre $\mathcal{N}$

- la *función de extracción de traza* $tr : Path^*_{\mathcal{N}} \cup Path^\omega_{\mathcal{N}} \to \Sigma^* \cup \Sigma\omega$ extrae los eventos de un camino dado y está dada por $tr(s) = \epsilon$, $tr(s \cdot a \cdot \rho) = a \cdot tr(\rho)$

- un estado $s \in S$ es *alcanzable* a partir del estado inicial si existe $\rho \in Path^*_{\mathcal{N}}$ que finalice en el estado s y denotamos $s_0 \stackrel{w}{\Rightarrow} s$ cuando s sea alcanzable mediante algún camino ρ tal que $tr(\rho) = w$

- el *lenguaje finito* $\mathcal{L}^*(\mathcal{N})$ y el *lenguaje infinito* $\mathcal{L}^\omega(\mathcal{N})$ asociados a $\mathcal{N}$ se definen como

$$\mathcal{L}^*(\mathcal{N}) = \{w \in \Sigma^* \mid \exists \rho \in Path^*_{\mathcal{N}} : tr(\rho) = w\}$$
$$\mathcal{L}^\omega(\mathcal{N}) = \{\sigma \in \Sigma^\omega \mid \exists \varrho \in Path^\omega_{\mathcal{N}} : tr(\varrho) = \sigma\}$$

- el *lenguaje* asociado al LTS $\mathcal{N}$ se define como

$$\mathcal{L}(\mathcal{N}) = \mathcal{L}^*(\mathcal{N}) \cup \mathcal{L}^\omega(\mathcal{N})$$

Sea $w \in \Sigma^*$, $\sigma \in \Sigma^\omega$, entonces:

- $|w|$ es la *longitud* de la traza

- sea $a \in \Sigma$ entonces $a \in w$ sí y sólo si $w \in \Sigma^* a \Sigma^*$ y $a \in \sigma$ sí y sólo si $\sigma \in \Sigma^* a \Sigma^\omega$

- sea $i \in \{1, \ldots, |w|\}$, w_i es el i-ésimo evento de la cadena w

- la *proyección observable* de w está dada por la función $obs : \Sigma^* \to \Sigma_o^*$ definida como

$$obs(w) = \begin{cases} \epsilon & \text{si } w = \epsilon \\ a \cdot obs(w') & \text{si } w = a \cdot w' \wedge a \in \Sigma_o \\ obs(w') & \text{si } w = a \cdot w' \wedge a \notin \Sigma_o \end{cases}$$

2.1.2. Restricciones sobre los modelos

Los LTS con los que trabajamos deben cumplir con algunas restricciones. Sea $\mathcal{N} = \langle S, \Sigma, \delta, s_0 \rangle$ un LTS, entonces $\mathcal{N}$ debe cumplir:

1. $\forall s \in S, (s, a, s') \in \delta, \quad$ para algunos $s' \in S, a \in \Sigma$
2. $\mathcal{L}^{\omega}(\mathcal{N}) \cap \Sigma^{*}\Sigma_{u}^{\omega} = \emptyset$

La primera condición indica que el sistema no puede llegar a un punto donde le sea imposible evolucionar, es decir, asumimos un sistema sin *deadlocks*. La segunda condición indica que el sistema no puede aceptar trazas que contengan ciclos compuestos únicamente por eventos inobservables.

Observación: La proyección observable definida en Sección 2.1.1 también se extiende a trazas infinitas, tomando el límite de sus prefijos finitos. Notar que el resultado de esta operación sobre trazas infinitas siempre resulta en una traza observable infinita, ya que exigimos que no haya ciclos de eventos inobservables en los modelos.

2.2. Diagnosticabilidad sobre LTS

La propiedad de **diagnosticabilidad** garantiza que toda ocurrencia de una falla en un sistema pueda ser detectada, ya que la continuación de cada traza que contenga una falla será en algún momento distinguible de otra traza que no tenga un evento de falla. Tanto las secuencias de estados como los eventos inobservables no están disponibles a la hora de decidir sobre la ocurrencia de una falla, lo cual debe inferirse únicamente mediante el análisis de las proyecciones observables de los eventos que ocurren en el sistema. La propiedad de diagnosticabilidad es una propiedad deseable para un sistema, ya que un sistema diagnosticable puede anticiparse o actuar en consecuencia a la ocurrencia de una falla, corrigiéndose o tomar medidas necesarias para mitigar los daños que la falla pueda llegar a ocasionar.

Un sistema se dice **no diagnosticable** si la ocurrencia de algún evento de falla no puede garantizarse a partir de ninguna serie de observaciones sobre el sistema, es decir, si existen dos trazas infinitas en el sistema cuya proyección observable es la misma, pero una de las trazas presenta una falla mientras que la otra no presenta eventos de falla.

Definición (Diagnosticabilidad)**.** *Sea $\mathcal{N}$ un LTS, decimos que $\mathcal{N}$ es* ***diagnosticable*** *sí y sólo si $\forall f \in \Sigma_f$ se cumple*

$$\forall \sigma_1, \sigma_2 \in \mathcal{L}^\omega(\mathcal{N}) : obs(\sigma_1) = obs(\sigma_2),\ si\ f \in \sigma_1\ entonces\ f \in \sigma_2$$

Ejemplo: Volviendo a la Figura 2, el lenguaje infinito asociado al LTS es $\mathcal{L}^\omega(\mathcal{M}) = (u + f)a^\omega$. Si tomamos las trazas $\sigma_1 = ua^\omega$ y $\sigma_2 = fa^\omega$, tenemos que $obs(\sigma_1) = obs(\sigma_2) = a^\omega$, $f \in \sigma_2$ pero $f \notin \sigma_1$, por lo tanto $\mathcal{N}$ no es diagnosticable.

2.2.1. Método "Twin-Plant"

Se han desarrollado distintos algoritmos para el chequeo de diagnosticabilidad de un sistema. A continuación detallamos el método "Twin-Plant" [12], el cual tiene la ventaja de tener una complejidad polinomial, en comparación a métodos previos con complejidad exponencial. Consiste en correr en paralelo dos LTS derivados a partir del LTS original, y verificar si existe un camino finalizado en algún ciclo que haga que uno de los LTS presente una falla que no se da en el otro LTS siguiendo un camino con las mismas observaciones.

El algoritmo es el siguiente:

Algoritmo 1: Sea $\mathcal{N} = \langle S, \Sigma, \delta, s_0 \rangle$ un LTS:

1. Construir el LTS $\mathcal{N}_o = \langle S_o, \Sigma_o, \delta_o, s_0^o \rangle$ de la siguiente manera:

   - $S_o = \{(s, l) \mid s \in S_1 \cup \{s_0\}, l \in \{N, F\}\}$ es el conjunto finito de estados, donde $S_1 = \{s \in S \mid \exists (s', a, s) \in \delta : a \in \Sigma_o\}$ es el conjunto de estados de $\mathcal{N}$ alcanzables por transiciones observables, y l es una etiqueta que indica si ha ocurrido un evento de falla entre s_0 y s,
   - Σ_o es el conjunto de eventos observables,
   - $\delta_o \subseteq S_o \times \Sigma_o \times S_o$ es el conjunto de transiciones. $((s, l), a, (s', l')) \in \delta_o$ sí y sólo si $s \overset{w}{\Rightarrow} s'$ para algún $w \in \Sigma_u^* a$ y

$$l' = \begin{cases} F & \text{si } f \in w \text{ para algún } f \in \Sigma_f \\ l & \text{c.c.} \end{cases}$$

- $s_0^o = (s_0, N) \in S_o$ es el estado inicial.

2. Computar $\mathcal{N}_d = \mathcal{N}_o \parallel \mathcal{N}_o$, es decir, la composición de $\mathcal{N}_o$ consigo mismo. $\mathcal{N}_d$ es un LTS dado por la 4-úpla $\langle S_d, \Sigma_o, \delta_d, s_0^d \rangle$, donde:

   - $S_d = \{(s_1^o, s_2^o) \mid s_1^o, s_2^o \in S_o\}$ es el conjunto de estados.
   - Σ_o es el conjunto de eventos.
   - $\delta_d \subseteq S_d \times \Sigma_o \times S_d$ es el conjunto de transiciones. En donde se cumple que $((s_1^o, s_2^o), a, (t_1^o, t_2^o)) \in \delta_d$ sí y sólo si (s_1^o, a, t_1^o) y (s_2^o, a, t_2^o) están en δ_o.
   - $s_0^d = (s_0^o, s_0^o) \in S_d$ es el estado inicial.

3. Verificar si en $\mathcal{N}_d$ se da que $s \stackrel{w}{\Rightarrow} s$, para $w \neq \epsilon$ y para algún estado $s = ((s_1, l_1), (s_2, l_2))$ tal que $l_1 \neq l_2$. Si esto ocurre, el sistema es no diagnosticable. En caso contrario el sistema es diagnosticable.

Notar que la presencia de dicho ciclo implica que existen dos trazas infinitas con la misma proyección observable en $\mathcal{N}$ tales que una de ellas presenta una falla que no ha ocurrido en la otra traza.

Ejemplo: Tomando nuevamente el LTS de la Figura 2, podemos aplicar el algoritmo presentado para ver por qué el sistema no es diagnosticable.

Siguiendo el primer paso del algoritmo, obtenemos $\mathcal{N}_o$, el cual se muestra en la Figura 3 y acepta solamente la traza a^*.

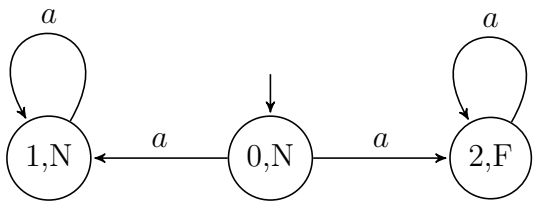


Figura 3: $\mathcal{N}_o$

La composición de $\mathcal{N}_o$ consigo mismo, $\mathcal{N}_d = \mathcal{N}_o \parallel \mathcal{N}_o$ es derivada en el segundo paso del algoritmo, y se muestra en la Figura 4.

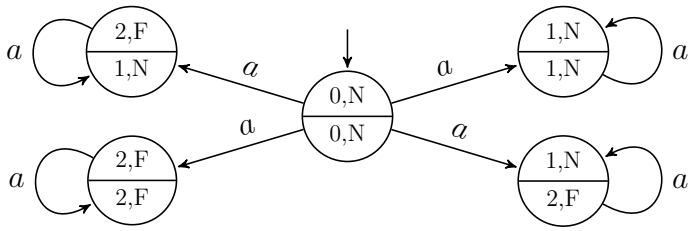


Figura 4: $\mathcal{N}_d$

Puede verse que existe un ciclo $((1,N),(2,F)) \stackrel{a}{\Rightarrow} ((1,N),(2,F))$ y otro ciclo $((2,F),(1,N)) \stackrel{a}{\Rightarrow} ((2,F),(1,N))$, por lo tanto el sistema $\mathcal{N}$ no es diagnosticable.

Observación: El último paso del algoritmo puede optimizarse eliminando los estados que contienen etiquetas idénticas, y verificar si en el sistema resultante hay ciclos. En la Figura 5 se observa la optimización del sistema de la Figura 4, con los estados y transiciones desvanecidas.

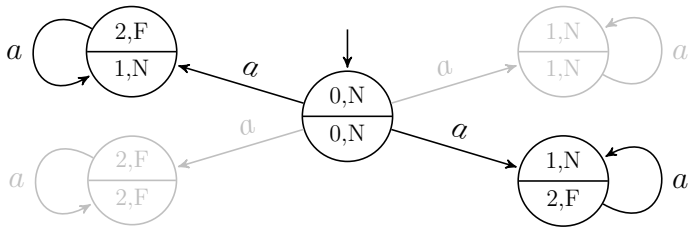


Figura 5: $\mathcal{N}_d$ optimizado

3. pLTS

3.1. Introducción a pLTS

El formalismo de sistemas de transiciones etiquetados ha sido aceptado como base para la especificación formal y verificación de sistemas. Dependiendo del tipo de información que se tenga sobre el sistema, el modelado del mismo se puede hacer más detallado para permitir un mejor análisis. La introducción de probabilidades es una de las posibles adiciones que se pueden incorporar a un modelo. En esta sección trabajamos sobre sistemas de transiciones etiquetados probabilistas, los cuales añaden a los sistemas de transiciones etiquetados la probabilidad de ocurrencia de las transiciones.

Definición (pLTS)**.** *Un Sistema de Transiciones Etiquetado Probabilístico (o pLTS por sus siglas en inglés) es una 4-úpla* $\mathcal{Q} = \langle S, \Sigma, \delta, s_0 \rangle$ *donde:*

- S *es un conjunto finito de estados,*
- Σ *es un conjunto finito de eventos,*
- $\delta : S \times \Sigma \times S \to [0, 1]$ *es la función de transición probablística,*
- s_0 *es el estado inicial.*

La función de transición debe cumplir que $\forall s \in S, \sum_{s' \in S, a \in \Sigma} \delta(s, a, s') = 1$, es decir, que las transiciones originadas en cada estado definan una *distribución de probabilidad* sobre el conjunto de estados.

Puede verse que un pLTS es un LTS cuyas transiciones están acompañadas por la probabilidad de ser ejecutadas en el siguiente paso.

Ejemplo: En la Figura 6 podemos observar un pLTS, el cual se representa de la misma manera que un LTS, con el agregado de la probabilidad de ocurrencia de cada transición, graficándose únicamente las transiciones con probabilidades mayores a 0.

3.1.1. Probabilidades de trazas y caminos en un pLTS

Dado un pLTS, podemos obtener un LTS redefiniendo δ como todas las triplas (s, a, s') tales que $\delta(s, a, s') > 0$. De esta manera, podemos redefinir

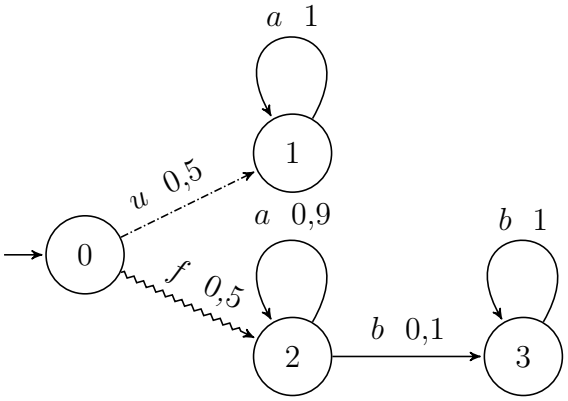


Figura 6: pLTS

todos los operadores vistos en la Sección 2.1.1 sobre pLTS utilizando su LTS asociado. Por lo tanto una traza será aceptada por un pLTS si es aceptada por su LTS asociado.

Al igual que para LTS, también requerimos que los pLTS cumplan con las restricciones de ausencia de ciclos inobservables y deadlocks definidas en la Sección 2.1.2.

La información sobre las probabilidades de las transiciones en un pLTS nos permite calcular qué tan probable es la ocurrencia de alguna traza o camino en particular. Sea $\mathcal{Q} = \langle S, \Sigma, \delta, s_0 \rangle$ un pLTS, entonces

- la *probabilidad de ocurrencia del camino* $\rho = s_0 \cdot a_1 \cdot s_1 \cdots a_n \cdot s_n \in Path^*_{\mathcal{Q}}$ está dada por
$$Pr(\rho) = \prod_{i=0}^{n-1} \delta(s_i, a_{i+1}, s_{i+1})$$
- la *probabilidad de ocurrencia de la traza* $w \in \mathcal{L}^*(\mathcal{Q})$ está dada por
$$Pr(w) = \sum_{\{\rho \in Path^*_{\mathcal{Q}} | tr(\rho) = w\}} Pr(\rho)$$

3.2. Diagnosticabilidad y Diagnosis en pLTS

Aprovechando el hecho de que en un pLTS las transiciones vienen acompañadas de su probabilidad de ocurrencia, el chequeo de diagnosticabilidad puede refinarse para dar más información: ahora podemos saber "qué tan no

diagnosticable" es un modelo dado. En particular, para una traza observable infinita, podemos obtener la probabilidad de que la misma sea una traza no diagnosticable (una traza en la cual no podemos decir si una falla ocurrió o no); o podemos calcular la probabilidad de que en una traza no diagnosticable ocurra una falla; entre otras cosas.

El análisis que hacemos en esta sección se basa en la construcción de un Diagnosticador, el cual es una máquina que captura el comportamiento observable del sistema y lleva información en sus estados sobre posibles ocurrencias de fallas, permitiendo también hacer un diagnóstico sobre la ocurrencia de dichas fallas. Aquí usamos la máquina con información sobre probabilidades en las transiciones presentada en [19], la cual se basa en el concepto del Diagnosticador introducido en [16].

3.2.1. Diagnosticador estocástico

El **Diagnosticador estocástico** es una máquina construida a partir de un pLTS que puede ser usada dinámica o estáticamente para describir el comportamiento de un sistema.

- Estáticamente usamos el Diagnosticador estocástico para analizar el grado de *diagnosticabilidad* del sistema, calculando algunos valores importantes como la probabilidad de ocurrencia de trazas no diagnosticables, la probabilidad de ocurrencia de fallas, la probabilidad de fallas dentro de trazas no diagnosticables, etc.

- Dinámicamente usamos el Diagnosticador estocástico para realizar *diagnosis* sobre el sistema, es decir, para conocer las siguientes informaciones:

    1. estimar el estado actual del sistema luego de observar cada evento,
    2. estimar la ocurrencia de fallas en el camino,
    3. calcular la probabilidad de cada una de las estimaciones anteriores.

3.2.2. Construcción del Diagnosticador estocástico

El Diagnosticador estocástico se construye en base a un pLTS, donde cada estado del Diagnosticador está compuesto por uno o más pares de elementos llamados **componentes**, formados cada uno por un estado del pLTS y una

etiqueta F o N para representar si ha ocurrido o no una falla en el sistema respectivamente.

Las transiciones entre dos estados se originan a través de un evento observable del sistema, y cada transición lleva información sobre las probabilidades de evolucionar entre cada componente de los estados de origen y destino.

Definición (Diagnosticador estocástico)**.** *Un Diagnosticador estocástico asociado a un pLTS* $\mathcal{Q} = \langle S, \Sigma, \delta, s_0 \rangle$ *es una 5-úpla* $\mathcal{D} = \langle Q_d, \Sigma_d, \delta_d, q_0, \Phi \rangle$*, donde*

- $Q_d \subseteq \mathcal{P}(S \times \{N, F\})$ *es el conjunto de estados, donde* N *y* F *son etiquetas que representan la ausencia u ocurrencia de fallas a lo largo del camino.*
- $\Sigma_d = \Sigma_o$ *es el conjunto de eventos, formado por los eventos observables de* $\mathcal{Q}$*.*
- $\delta_d : Q_d \times \Sigma_d \to Q_d$ *es la función de transición entre estados, definida luego.*
- $q_0 = \{(s_0, N)\}$ *es el estado inicial.*
- $\Phi : Q_d \times \Sigma_d \to [0, 1]^{n \times m}$ *es la función de probabilidad de transiciones para los componentes de cada estado, definida luego.*

Aquí $[0, 1]^{n \times m}$ representa al conjunto de matrices finitas cuyos elementos están en el intervalo $[0, 1]$.

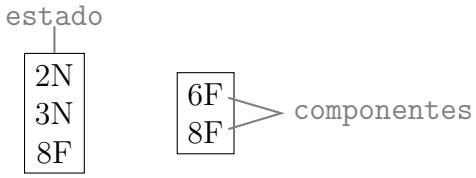


Figura 7: Diferencia entre estados y componentes

Llamamos estados **ambiguos** a aquellos estados que contienen al menos una componente etiquetada con N y otra etiquetada con F. En la Figura 7, el estado de la izquierda es ambiguo mientras que el de la derecha no lo es. También diremos que una componente es ambigua si está dentro de un estado ambiguo. La cantidad de componentes de un estado q se denota con $|q|$.

Para poder definir δ_d, definimos primero la función de propagación de etiquetas $LP : \Sigma^* \times \{N, F\} \to \{N, F\}$, la cual está dada por

$$LP(w, l) = \begin{cases} N & \text{si } l = N \wedge \forall f \in \Sigma_f, f \notin w \\ F & \text{c.c.} \end{cases}$$

Esta función etiqueta con F cuando se ejecuta una traza que contiene una falla o propaga una F ya existente. Notar que una vez que la función LP añade la etiqueta F, la misma no puede ser removida por futuras aplicaciones de la misma función.

Usando esta función, definimos la función de transición del Diagnosticador estocástico de la siguiente manera:

$$\delta_d(q, a) = \bigcup_{(s,l)\in q} \{(s', l') \mid \exists w \in \Sigma_u^* a : s \stackrel{w}{\Rightarrow} s' \wedge l' = LP(w, l)\}$$

Para construir el conjunto de matrices asociadas a Φ es necesario imponer un orden a las componentes de cada estado. Este orden puede ser elegido arbitrariamente, pero por convención denotamos con $comp_{q,i}$ a la i-ésima componente de un estado q.

Sean $(s_i, l_i) = comp_{q,i}$ y $(s_j, l_j) = comp_{\delta_d(q,a),j}$ para algún $q \in Q_d$ y $a \in \Sigma_d$ entonces definimos al elemento i, j de la matriz asociada a $\Phi(q, a)$ como

$$\Phi_{i,j}(q, a) = \sum_{\{w\in\Sigma_u^* a \,|\, s_i \stackrel{w}{\Rightarrow} s_j \ \wedge \ l_j = LP(w,l_i)\}} Pr(w)$$

El tamaño de la matriz asociada a $\Phi(q, a)$ es $|q|\times|\delta_d(q, a)|$. El elemento i, j de la matriz asociada a $\Phi(q, a)$ representa la probabilidad de que al observarse un evento a el sistema evolucione desde la componente i-ésima del estado q hacia la componente j-ésima del estado dado por $\delta_d(q, a)$.

Gráficamente representamos al Diagnosticador como un LTS cuyos estados son rectangulares para diferenciarlos de los del modelo. Por cada transición, hay un evento observable y una matriz de probabilidades asociadas a ella.

Ejemplo: En la Figura 8 vemos el Diagnosticador estocástico asociado al pLTS de la Figura 6. El mismo se compone de los estados $q_0 = \{(0, N)\}$, $q_1 = \{(1, N), (2, F)\}$ y $q_2 = \{(3, F)\}$. Si el Diagnosticador se encuentra en su estado inicial y se observa un evento `a`, entonces el estado actual del

Diagnosticador pasa a ser q_1. Esta transición se traduce en el sistema original de la Figura 6 como que a partir del estado inicial han ocurrido una serie de eventos no observables seguidos de un evento `a`, y a partir de ello el sistema puede evolucionar tanto al estado 1 sin que ocurra ninguna falla en el camino, como así también al estado 2 con la ocurrencia de alguna falla en el camino. En ese momento no puede saberse exactamente si una falla ha ocurrido o no en el sistema, por eso decimos que el estado q_1 es ambiguo.

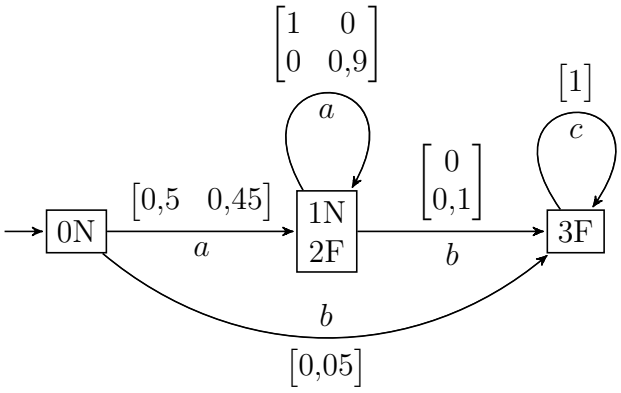


Figura 8: Diagnosticador estocástico asociado al pLTS de la Figura 6

Como dijimos previamente, el ejercicio de determinar si una falla ha ocurrido en el sistema a partir de observaciones sobre el mismo se conoce como diagnosis del sistema, y lo detallamos en la Sección 3.2.5.

3.2.3. Diagnosticabilidad usando el Diagnosticador estocástico

Al igual que en un LTS, un pLTS es diagnosticable cuando no existen dos trazas infinitas con la misma proyección observable tal que sólo una de ellas presenta una falla, es decir, cuando cumple que

$$\forall \sigma_1, \sigma_2 \in \mathcal{L}^{\omega}(\mathcal{Q}) : obs(\sigma_1) = obs(\sigma_2), \text{ si } f \in \sigma_1 \text{ entonces } f \in \sigma_2$$

Para chequear si se cumple esta propiedad basta con considerar al LTS asociado al pLTS en cuestión y aplicar el método "Twin-Plant". Sin embargo, de esta manera obtenemos una respuesta afirmativa o negativa con respecto a la diagnosticabilidad del sistema, desaprovechando por completo la información que nos brindan las probabilidades introducidas al modelo. Si el sistema es diagnosticable, el método "Twin-Plant" asegura que toda

falla será diagnosticada correctamente en algún momento. Sin embargo si el sistema resulta no diagnosticable, una única traza alcanza como contraejemplo para catalogar a todo el sistema como no diagnosticable, por más que la probabilidad de ocurrencia de dicha traza sea casi nula. Intentando no dar una respuesta tan taxativa, en esta sección presentamos un análisis sobre sistemas no diagnosticables que aprovecha la información de las probabilidades en las transiciones del modelo para analizar "que tan no diagnosticable" es el sistema. Para ello nos enfocamos en algunas probabilidades referidas a observaciones infinitas sobre el sistema.

Sea $\mathcal{Q} = \langle S, \Sigma, \delta, s_0 \rangle$ un pLTS, sea $f \in \Sigma_f$ y sea $\beta = obs(\alpha_0)$ para algún $\alpha_0 \in \mathcal{L}^\omega(\mathcal{Q})$, queremos analizar el resultado de los siguientes valores:

$p_{\neg d}$ la probabilidad de que exista una traza infinita no diagnosticable cuya proyección observable sea β, es decir

$$p_{\neg d} = Pr(\exists \alpha_1, \alpha_2 \in \mathcal{L}(\mathcal{Q}) : obs(\alpha_1) = obs(\alpha_2) = \beta \wedge (f \in \alpha_1 \veebar f \in \alpha_2))^1$$

p_F la probabilidad de que exista una traza cuya proyección observable sea β y presente una falla, es decir

$$p_F = Pr(\exists \alpha_1 \in \mathcal{L}(\mathcal{Q}) : obs(\alpha_1) = \beta \wedge f \in \alpha_1)$$

$p_{F|Nd}$ la probabilidad de que ocurra una falla en una traza cuya proyección observable es β, dado que es una traza no diagnosticable, es decir

$$p_{F|Nd} = Pr(\exists \alpha_1 \in \mathcal{L}(\mathcal{Q}) : f \in \alpha_1 \wedge obs(\alpha_1) = \beta \mid \exists \alpha_2 \in \mathcal{L}(\mathcal{Q}) : obs(\alpha_2) = \beta \wedge f \notin \alpha_2)$$

El Diagnosticador estocástico puede utilizarse para calcular estos valores: en el trabajo de [15] se parte de un Diagnosticador parecido al desarrollado aquí (difiriendo en la representación de sus estados), y a partir de él se extrae una cadena de Markov para realizar un análisis asintótico sobre las trazas infinitas que recorren la cadena. A continuación desarrollamos un método similar aplicado al Diagnosticador estocástico.

Una cadena de Markov [10] es un tipo especial de proceso estocástico discreto en el que la probabilidad de que ocurra un evento depende únicamente del evento inmediatamente anterior. Esta característica de falta de memoria recibe el nombre de propiedad de Markov.

[1] El símbolo $\veebar$ hace referencia al operador lógico de disyunción exclusiva.

Definición (DTMC)**.** *Una* ***Cadena de Markov de Tiempo Discreto*** *(DTMC por sus iniciales en inglés) es una 3-úpla* $\mathcal{C} = \langle S, \delta, s_0 \rangle$*, donde:*

- S *es un conjunto numerable de estados,*
- $s_0 \in S$ *es el estado inicial,*
- $\delta : S \times S \to [0, 1]$ *es una función de transición probabilista.*

Estas cadenas pueden pensarse como procesos que se mueven a través de un conjunto de estados $s_1, s_2, \ldots$ en el cual, si el proceso se encuentra en el estado s_i, puede moverse al estado s_j con probabilidad p_{ij}. Cuando el número de estados es finito, la función de transición probabilista suele ser expresada como una matriz de transiciones, donde cada fila es la distribución de probabilidad de cada estado. Una cadena de Markov finita puede verse también como un pLTS sin etiquetas en sus transiciones.

A pesar de que el Diagnosticador estocástico tiene matrices de probabilidades asociadas a cada transición, su estructura esconde una cadena de Markov cuyos estados son cada una de las componentes de cada estado del Diagnosticador. La existencia de esta cadena de Markov nos permite hacer uso de las técnicas existentes para hacer un análisis sobre la diagnosticabilidad del sistema.

Para describir esta cadena, hacemos uso de la función $\Omega : Q_d \times Q_d \to [0, 1]^{n \times m}$ que se aplica a dos estados de un Diagnosticador y está dada por

$$\Omega(q_i, q_j) = \sum_{\{a \in \Sigma_d \,|\, \delta_d(q_i, a) = q_j\}} \Phi(q_i, a)$$

La matriz que resulta de la operación $\Omega(q_i, q_j)$ es la matriz dada por la suma de todas las matrices asociadas a las transiciones entre los estados q_i y q_j del Diagnosticador. En caso de que no exista ninguna transición entre dichos estados, el resultado de $\Omega(q_i, q_j)$ será la matriz compuesta de ceros de tamaño $|q_i| \times |q_j|$.

Definición (DTMC asociada a un Diagnosticador estocástico)**.** *Sea* $\mathcal{Q} = \langle S, \Sigma, \delta, s_0 \rangle$ *un pLTS y* $\mathcal{D} = \langle Q_d, \Sigma_d, \delta_d, q_0, \Phi \rangle$ *el Diagnosticador estocástico asociado a* $\mathcal{Q}$*, entonces la cadena de Markov asociada a* $\mathcal{D}$ *es* $\Pi(\mathcal{D}) = \langle S^\Pi, \delta^\Pi, s_0^\Pi \rangle$*, donde:*

- $S^\Pi \subseteq Q_d \times S \times \{N, F\}$ *es el conjunto de todas las componentes de* $\mathcal{D}$*, donde si* $(q, s, l) \in S^\Pi$ *entonces* $(s, l) \in q$

- δ^{Π} *está dada por la siguiente matriz de transiciones:*

$$tr = \begin{bmatrix} \Omega(q_0, q_0) & \cdots & \Omega(q_0, q_n) \\ \vdots & \ddots & \vdots \\ \Omega(q_n, q_0) & \cdots & \Omega(q_n, q_n) \end{bmatrix}$$

- $s_0^{\Pi} = (q_0, s_0, N)$ *es el estado inicial.*

Inferimos un órden sobre el conjunto de estados de la cadena, y usamos estados como índices de la matriz de transiciones.

Ejemplo: En nuestro Diagnosticador de ejemplo de la Figura 8, la cadena de Markov asociada $\Pi(\mathcal{D})$ tiene un conjunto de estados $\{(q_0, 0, N), (q_1, 1, N), (q_1, 2, F), (q_2, 3, F)\}$, su estado inicial es $(q_0, 0, N)$ y cuyas transiciones entre estados están dadas por la siguiente matriz:

$$tr = \begin{array}{c} (q_0, 0, N) \\ (q_1, 1, N) \\ (q_1, 2, F) \\ (q_2, 3, F) \end{array} \begin{bmatrix} 0 & 0{,}5 & 0{,}45 & 0{,}05 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0{,}9 & 0{,}1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.2.4. Comportamiento asintótico de la cadena de Markov

El hecho de utilizar cadenas de Markov para nuestro análisis de diagnosticabilidad nos permite aprovechar algunas de sus propiedades [10]. Sea $\mathcal{C} = \langle S, \delta, s_0 \rangle$ un DTMC, y sean $x, y \in S$, entonces:

- denotamos con ρ_{xy} a la probabilidad de que a partir del estado x, en algún momento en el futuro se alcance el estado y,
- un estado $x \in S$ es llamado **recurrente** si $\rho_{xx} = 1$, y en caso contrario es llamado **transitorio**,
- todo estado transitorio deja de ser visitado en algún momento en el futuro,
- si un estado recurrente es visitado una vez, entonces será visitado infinitas veces en el futuro,

- si el DTMC es finito, entonces debe haber al menos un estado recurrente,
- si $\rho_{xy} > 0$, entonces y es **alcanzable** a partir de x, y lo denotamos con $x \to y$,
- si x es un estado recurrente y $x \to y$, entonces y también es recurrente y $y \to x$,
- un conjunto de estados recurrentes $\{x_1, x_2, \ldots, x_n\}$ tal que $x_i \to x_j$ y $x_j \to x_i$ $\forall i, j \in \{1, \ldots, n\}$ es llamado una **clase de recurrencia**,
- $\mathcal{C}$ es llamado **irreducible** si contiene solo una clase de recurrencia.

Notar que si el sistema tiene al menos una transición etiquetada con un elemento de Σ_f, entonces $\Pi(\mathcal{D})$ no puede ser irreducible, ya que desde el estado inicial se puede alcanzar algún estado que tenga alguna componente etiquetada con F, pero a partir de ese estado nunca se puede volver al estado inicial, ya que el mismo no posee ninguna componente etiquetada con F.

Estudiando el comportamiento asintótico de la cadena (i.e., cuando el sistema se estabiliza) podemos calcular probabilidades relevantes a las trazas observables infinitas del sistema. El hecho de considerar trazas infinitas no es un problema en la práctica, puesto que también calculamos el valor avg_{steps}, el cual representa el número promedio de observaciones necesarias para que el sistema converja a una etapa donde la diagnosticabilidad de la traza ya está definida.

Para calcular los valores para $p_{\neg d}$, p_F, $p_{F|Nd}$, $p_{\neg F|Nd}$ y avg_{steps} procedemos de la siguiente manera:

1. Clasificar los estados de $\Pi(\mathcal{D})$, extrayendo el conjunto de estados recurrentes $\{\lambda_1, \ldots, \lambda_k\}$ agrupados en las clases de recurrencia $\{\zeta_1, \ldots, \zeta_h\}$, y el conjunto de estados transitorios $\{\nu_1, \ldots, \nu_r\}$.
2. Reorganizar las filas y columnas de la matriz de transiciones de $\Pi(\mathcal{D})$ en su *forma canónica*, es decir, reorganizar la matriz de tal forma que los estados transitorios están en las primeras posiciones y los estados de cada clase de recurrencia están juntos. Obtenemos así una matriz de transiciones de la forma:

$$tr = \begin{bmatrix} Q & R \\ 0 & C \end{bmatrix}, \qquad C = \begin{bmatrix} tr_{\zeta_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & tr_{\zeta_h} \end{bmatrix}$$

donde cada tr_{ζ_i} es una matriz de transiciones de la clase de recurrencia ζ_i. La matriz $C \in [0,1]^{k \times k}$ contiene probabilidades de transiciones entre estados recurrentes, la matriz $R \in [0,1]^{r \times k}$ contiene probabilidades de transiciones desde estados transitorios hacia estados recurrentes, y la matriz $Q \in [0,1]^{r \times r}$ contiene probabilidades de transiciones entre estados transitorios.

3. Computamos la matriz fundamental de la cadena de Markov, dada por $N = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1}$, donde I es la matriz identidad de tamaño $r \times r$. Sumando la i-ésima fila de esta matriz obtenemos el número promedio de pasos posibles antes de alcanzar una clase de recurrencia, comenzando desde el estado transitorio i. Usando esta matriz, podemos computar la matriz $B = N \cdot R$. El elemento (i,j)-ésimo de esta matriz contiene la probabilidad de ser absorbido en el estado recurrente j a partir del estado transitorio i.

4. Sea $\zeta_{\neg d}$ el subconjunto de clases de recurrencia que contienen solo estados asociados a componentes ambiguos del Diagnosticador, y sea ζ_F el subconjunto de clases de recurrencia que contienen solo estados asociados a componentes marcados con F en el Diagnosticador, entonces con ayuda de las matrices N y B podemos calcular los siguientes valores:

   - $p_{\neg d}$ es la probabilidad de ser absorbido en una de las clases de $\zeta_{\neg d}$ a partir del estado inicial. Está dada por

   $$p_{\neg d} = \sum_{\zeta \in \zeta_{Nd}} \sum_{\lambda \in \zeta} B_{s_0^{\Pi}, \lambda}$$

   - p_F es la probabilidad de ser absorbido en una de las clases de ζ_F a partir del estado inicial. Está dada por

   $$p_F = \sum_{\zeta \in \zeta_F} \sum_{\lambda \in \zeta} B_{s_0^{\Pi}, \lambda}$$

- $p_{F\wedge\neg d}$ es la probabilidad de ser absorbido en una de las clases pertenecientes tanto a $\zeta_{\neg d}$ como a ζ_F a partir del estado inicial. Usando la fórmula de Bayes calculamos el valor de $p_{F|\neg d}$ como

$$p_{F|\neg d} = \frac{p_{F\wedge\neg d}}{p_{\neg d}} = \frac{\sum\limits_{\zeta\in(\zeta_{\neg d}\cap\zeta_F)}\sum\limits_{\lambda\in\zeta} B_{s_0^{\Pi},\lambda}}{\sum\limits_{\zeta\in\zeta_{\neg d}}\sum\limits_{\lambda\in\zeta} B_{s_0^{\Pi},\lambda}}$$

- $p_{\neg F|\neg d}$ se obtiene de manera similar, sólo que ahora nos interesan las clases pertenecientes a $\zeta_{\neg d}$ pero no a ζ_F, es decir

$$p_{\neg F|\neg d} = \frac{\sum\limits_{\zeta\in(\zeta_{\neg d}-\zeta_F)}\sum\limits_{\lambda\in\zeta} B_{s_0^{\Pi},\lambda}}{\sum\limits_{\zeta\in\zeta_{\neg d}}\sum\limits_{\lambda\in\zeta} B_{s_0^{\Pi},\lambda}} = 1 - p_{F|\neg d}$$

- avg_{step} es la cantidad promedio de pasos necesarias para llegar a alguna clase de recurrencia a partir del estado inicial. Está dada por

$$avg_{step} = \sum_{i=1}^{r} N_{s_0^{\Pi},\nu_i}$$

Ejemplo: Procedemos a aplicar este algoritmo al DTMC extraído del Diagnosticador de la Figura 8:

1. Los estados transitorios son $\{(q_0, 0, N), (q_1, 2, F)\}$, y las clases de recurrencia son $\{(q_1, 1, N)\}$ y $\{(q_2, 3, F)\}$

2. La matriz de transiciones en forma canónica resulta:

$$tr = \begin{matrix}(q_0,0,N)\\(q_1,2,F)\\(q_1,1,N)\\(q_2,3,F)\end{matrix}\begin{bmatrix}0 & 0{,}45 & 0{,}5 & 0{,}05\\ 0 & 0{,}9 & 0 & 0{,}1\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

3. Las matrices $N = (I - Q)^{-1}$ y $B = N \cdot R$ resultan:

$$N = \begin{bmatrix}1 & 4{,}5\\ 0 & 10\end{bmatrix}, \qquad B = \begin{bmatrix}0{,}5 & 0{,}5\\ 0 & 1\end{bmatrix}$$

4. $\zeta_{\neg d} = \{(q_1, 1, N)\}$ y $\zeta_F = \{(q_2, 3, F)\}$, por lo que los valores sobre la diagnosticabilidad del pLTS de la Figura 6 resultan:

- $p_{\neg d} = 0{,}5$
- $p_F = 0{,}5$
- $p_{F \mid \neg d} = 0$
- $p_{\neg F \mid \neg d} = 1$
- $avg_{step} = 5{,}5$

Esto significa que luego de observar 6 eventos del sistema, hay un 50 % de probabilidad que la traza observada sea no diagnosticable. Sin embargo, todas las trazas no diagnosticables están libres de ocurrencias de fallas cuando consideramos trazas infinitas (dado que $p_{F \mid \neg d} = 0$). Estos resultados pueden verse a simple vista en el sistema original, ya que el modelo sólo está compuesto de pocos estados y transiciones. La utilidad de este análisis se hace más evidente a medida que el modelo crece en tamaño.

3.2.5. Diagnosis usando el Diagnosticador estocástico

Aparte del análisis sobre diagnosticabilidad, podemos utilizar el Diagnosticador para realizar diagnosis sobre el sistema. La diagnosis del sistema consiste en estimar en todo momento valores para el estado actual del sistema, la ocurrencia de fallas en el camino y la probabilidad de que la estimación sea correcta, a medida que se van observando nuevos eventos en el sistema.

Pueden existir varias trazas cuyas proyecciones observables coinciden. Por cada una de estas trazas, el estado inicial del sistema puede llegar a evolucionar hacia distintos estados. El Diagnosticador estocástico representa esto evolucionando hacia un estado con (posiblemente) varias componentes siguiendo una traza observable. Cada una de estas componentes se corresponden con un posible estado actual del sistema original y una etiqueta que informa la posible ocurrencia de fallas en el camino.

Como el lenguaje asociado a cada pLTS es infinito (debido a las restricciones presentadas en la Sección 2.1.2), es imposible listar mediante el Diagnosticador estocástico todos las triplas *(estado actual, etiqueta, probabilidad)* para todas las proyecciones observables de trazas de un pLTS. Sin embargo, el Diagnosticador estocástico nos permite calcular estos valores a pesar de ser una máquina finita.

Supongamos que $w = a_1 \dots a_n \in \Sigma_o^*$ es el comportamiento observado del sistema, y que $(q_i)_{0 \leq i \leq n}$ es la secuencia de estados del Diagnosticador asociada a w. Entonces, el vector de ocurrencia de cada componente del estado q_n alcanzado por w está dado por

$$\phi(w) = \frac{\phi_{un}(w)}{\|\phi_{un}(w)\|}$$

donde $\phi_{un}(w)$ se obtiene del producto de matrices

$$\phi_{un}(w) = \prod_{i=0}^{n-1} \Phi(q_i, a_{i+1})$$

y $\|\phi_{un}(w)\|$ es la suma de todos los elementos del vector $\phi_{un}(w)$ (es decir, normalizamos el vector, para que el resultado sea una distribución de probabilidad).

Este resultado nos brinda un método para realizar la diagnosis del sistema calculando los vectores de probabilidad a partir de las matrices de probabilidad del Diagnosticador estocástico.

Ejemplo: Si en nuestro Diagnosticador de la Figura 8 observamos la cadena aaa, entonces el conjunto de componentes alcanzado es $\{(1N), (2F)\}$. Para obtener las probabilidades de cada componente, calculamos primero el vector sin normalizar:

$$\phi_{un}(aaa) = \begin{bmatrix} 0{,}5 & 0{,}45 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0{,}9 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0{,}9 \end{bmatrix} = \begin{bmatrix} 0{,}5 & 0{,}3645 \end{bmatrix}$$

luego lo normalizamos, obteniendo:

$$\phi(aaa) = \frac{[0{,}5 \quad 0{,}3645]}{0{,}5 + 0{,}3645} = [\overbrace{0{,}5784}^{1N} \quad \overbrace{0{,}4216}^{2F}]$$

por lo que luego de observar la cadena aaa, los posibles estados actuales son 1 y 2, la probabilidad de estar en el estado 1 y que no haya ocurrido una falla en el camino es de 0,5784 y la probabilidad de estar en el estado 2 y que haya ocurrido alguna falla en el camino es de 0,4216.

3.2.6. Análisis de los valores de Diagnosticabilidad calculados

A diferencia de los métodos tradicionales, los cuales se enfocan en brindar una respuesta binaria al problema de diagnosticabilidad, mediante el método que acabamos de presentar podemos cuantificar el grado de diagnosticabilidad de un sistema. Un sistema puramente diagnosticable debe tener una probabilidad nula de ocurrencia de trazas no diagnosticables. En caso contrario será no diagnosticable, y su nivel de no diagnosticabilidad estará dado por la probabilidad de la ocurrencia de las trazas no diagnosticables.

Además de eso, podemos también obtener otros valores interesantes, como la probabilidad de ocurrencia de fallas y la cantidad de observaciones necesarias para converger a un estado en el cual las trazas del sistema sean o no diagnosticables.

En la práctica, estos valores son útiles para la toma de decisiones sobre posibles adaptaciones a realizarse en sistemas no diagnosticables, o tolerar sistemas no diagnosticables en los cuales las trazas suficientemente largas tienden a una certeza de poseer o no una falla.

En sistemas críticos podría ser inaceptable la ocurrencia de trazas no diagnosticables, por más que esta probabilidad sea baja, mientras que en sistemas más flexibles podría llegar a aceptarse la ocurrencia de trazas no diagnosticables, siempre que su probabilidad sea baja. Por lo cual el tipo de sistema analizado tiene mucho que ver con el análisis de los valores calculados.

Podría darse el caso en que la probabilidad de ocurrencia de fallas en trazas no diagnosticables sea muy alta, pudiendo inferirse con alta certeza la ocurrencia de fallas en estas trazas. De manera similar, cuando la probabilidad de ocurrencia de fallas en trazas no diagnosticables es muy baja, la ocurrencia de una traza no diagnosticable implica una poca probabilidad de ocurrencia de una falla.

Entonces el **peor caso posible** se presenta cuando la ocurrencia de una traza no diagnosticable brinda el mayor grado de incertidumbre, es decir, cuando la probabilidad de ocurrencia de una falla en una traza no diagnosticable toma valores cercanos al 50 %. Un sistema que evidencie este caso debería ser reformulado para optimizar el nivel de certeza sobre la ocurrencia de fallas.

4. MDP

4.1. Introducción a MDP

Otro tipo de sistemas probabilísticos están dados por sistemas que exhiben comportamientos probabilísticos y no determinísticos, en los cuales la evolución entre dos estados se da en dos etapas: primero se elige no determinísticamente un evento a ejecutar sobre un conjunto de eventos disponibles, y luego ese evento provoca la evolución del estado actual hacia el siguiente estado conforme a una distribución de probabilidad. El no determinismo en los sistemas sirve para modelar varias situaciones, entre ellas:

- *entorno desconocido:* si el sistema interactúa con otros componentes cuyo comportamiento es desconocido, ésto puede ser modelado con el no determinismo.
- *concurrencia:* en un sistema distribuido compuesto por varias partes operando en paralelo, el no determinismo se usa para representar como se pueden intercalar las ejecuciones de cada parte del sistema.
- *subespecificación:* si ciertas partes de un sistema son desconocidas o muy complejas como para ser modeladas de manera eficiente, éstas puedes ser abstraídas usando no determinismo.

A su vez, podemos utilizar el no determinismo para capturar las formas posibles en las que un controlador puede influenciar el comportamiento del sistema. Por ende, las técnicas desarrolladas más adelante pueden verse como una síntesis de controladores sobre el sistema para que cumpla con las especificaciones requeridas.

Estos sistemas suelen representarse con **Procesos de decisión de Markov** [8], los cuales son muy estudiados y utilizados en áreas que van desde la robótica hasta la economía.

Definición (MDP)**.** *Un Proceso de Decisión de Markov (o MDP por sus iniciales en inglés) es una 4-úpla* $\mathcal{M} = \langle S, \Sigma, \delta, s_0 \rangle$ *donde:*

- S *es un conjunto finito de estados,*
- Σ *es un conjunto finito de eventos,*
- $\delta : S \times \Sigma \to Dist(S)$ *es una función parcial de transiciones probabilista,*

- s_0 *es el estado inicial.*

En la definición, $Dist(S)$ denota al conjunto de todas las distribuciones de probabilidad sobre S, donde una distribución de probabilidad (discreta) sobre un conjunto enumerable S es una función $\mu : S \to [0, 1]$ tal que $\sum_{s \in S} \mu(s) = 1$. Con $[s_0 \mapsto p_0, \ldots, s_n \mapsto p_n]$ representamos a la distribución que elige al elemento s_i con probabilidad p_i, y a todo otro elemento no incluido entre los s_i con probabilidad 0. Una distribución de la forma $[s_i \mapsto 1]$ es llamada distribución de punto.

Tal como sucede en sistemas modelados con LTSs y pLTSs, el conjunto de eventos de los sistemas modelados con MDP también está particionado en eventos observables e inobservables, con eventos de fallas entre los eventos inobservables.

Las transiciones entre estados ocurren en dos pasos: primero se hace una elección entre alguno de los eventos de Σ. El conjunto de eventos habilitados para un estado $s \in S$ se define como $A(s) \stackrel{\text{def}}{=} \{a \in \Sigma \mid \delta(s, a) \text{ está definido}\}$. La elección de algún $a \in A(s)$ es *no determinista*. Luego, un estado sucesor s' se elige al azar, siguiendo la distribución de probabilidad dada por $\delta(s, a)$.

4.1.1. Representación gráfica

La representación gráfica de un MDP es parecida a la de los modelos vistos previamente, salvo que ahora las transiciones están divididas en dos partes: por un lado está el evento que origina la transición y por otro lado está el conjunto de estados a los que la distribución de probabilidad asociada nos permite llegar. Para representar estas transiciones, las flechas se ramifican a la mitad del trayecto hacia cada uno de los estados alcanzables por la transición, y cada rama está acompañada de sus probabilidad de ocurrencia (las ramas de probabilidad nula no se grafican). La Figura 9 ejemplifica la representación de un MDP.

4.1.2. Caminos, trazas y lenguajes en MDP

Sea $\mathcal{M} = \langle S, \Sigma, \delta, s_0 \rangle$ un MDP, entonces:

- un *camino finito* sobre el MDP $\mathcal{M}$ es una secuencia $\rho = s_0 \cdot a_1 \cdot s_1 \cdots a_n \cdot s_n$ finalizada siempre en algún estado, donde $s_i \in S$, $a_i \in A(s_{i-1})$ y $\forall i \in \{0, \ldots, n-1\}, \delta(s_i, a_{i+1})(s_{i+1}) > 0$, y con $Path^*_{\mathcal{M}}$ denotamos todos los caminos finitos sobre $\mathcal{M}$,

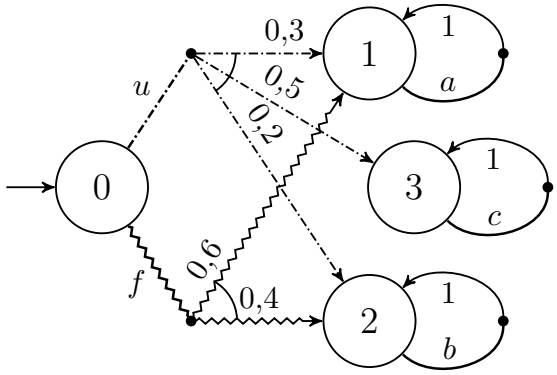


Figura 9

- un *camino infinito* sobre el MDP $\mathcal{M}$ es una secuencia $\varrho = s_0 \cdot a_1 \cdot s_1 \cdots$ donde $s_i \in S$, $a_i \in A(s_{i-1})$ y $\forall i \in \mathbb{N}, \delta(s_i, a_{i+1})(s_{i+1}) > 0$, y con $Path^{\omega}_{\mathcal{M}}$ denotamos todos los caminos infinitos sobre $\mathcal{M}$,
- sea $\rho = s_0 a_1 s_1 \dots a_n s_n \in Path^{*}_{\mathcal{M}}$, entonces $last(\rho) = s_n$,
- la *función de extracción de traza tr*, el concepto estado *alcanzable* y los lenguajes $\mathcal{L}^{*}(\mathcal{M})$, $\mathcal{L}^{\omega}(\mathcal{M})$ y $\mathcal{L}(\mathcal{M})$ se definen de manera similar a como fueron definidas en la Sección 2.1.1.

Al igual que LTS y pLTS, los modelos con los cuales trabajamos tienen restricciones similares a las de la Sección 2.1.2: pedimos que $\mathcal{L}^{\omega}(\mathcal{M}) \cap \Sigma^{*}\Sigma^{\omega}_{u} = \emptyset$ (i.e., no existen ciclos de eventos inobservables en $\mathcal{M}$) y asumimos que $\forall s \in S, A(s) \neq \emptyset$ para prevenir *deadlocks*.

4.1.3. Estrategias

El no determinismo puede ser muy útil para representar distintos comportamientos de los sistemas. Sin embargo, si queremos hacer un análisis probabilista sobre las propiedades del MDP necesitamos alguna forma de determinizar el modelo. Una **estrategia** es la encargada de realizar esta tarea, y para lograrlo elige un evento en cada estado del MDP basándose (o no) en las decisiones tomadas hasta el momento.

Definimos entonces una estrategia asociada a un MDP $\mathcal{M} = \langle S, \Sigma, \delta, s_0 \rangle$ como una función $\pi : Path^{*}_{\mathcal{M}} \to Dist(\Sigma)$ tal que $\pi(\rho)(a) > 0$ sólo si $a \in A(last(\rho))$.

Por lo general, la elección de un evento puede ser hecha al azar y depender del historial completo de un MDP, pero siempre tiene que estar limitada a eventos disponibles en el estado actual. El conjunto de todas las estrategias posibles para un MDP $\mathcal{M}$ se denota con $Str_{\mathcal{M}}$.

Hay muchas clases de estrategias [8] incluidas en $Str_{\mathcal{M}}$. Una estrategia es *determinista* si $\pi(\rho)$ es una distribución de punto, para todo $\rho \in Path_{\mathcal{M}}$, y una estrategia es *aleatoria* en caso contrario. Una estrategia tiene *memoria finita* si sólo hace uso de tramos finitos de los caminos para tomar decisiones. En particular, si la estrategia solamente tiene en cuenta el estado actual para la toma de decisiones, se dice que es una **estrategia sin memoria**, y está dada por $\pi : S \to Dist(\Sigma)$.

Bajo una estrategia definida, el comportamiento de un MDP es totalmente probabilista, y puede ser capturado por una cadena de Markov posiblemente infinita. En los casos en que la estrategia elegida es una estrategia sin memoria, al aplicar la estrategia a un MDP podemos obtener un pLTS que respeta la estructura del MDP que determiniza. El análisis que hacemos en este trabajo se enfoca en obtener únicamente estrategias sin memoria. Esta decisión fue tomada por los siguientes motivos:

- Mediante una estrategia sin memoria podemos determinizar un MDP de tal manera que obtenemos un pLTS que respeta fielmente la estructura original del MDP, ya que no necesitamos agregar nuevos estados.
- Creemos que debido al tipo de análisis que realizamos, el uso de cualquier estrategia puede reducirse a una estrategia sin memoria que otorgue resultados similares. En la Sección 4.2.7 presentamos un esquema de la prueba.

4.1.4. pLTS derivado de un MDP con una estrategia sin memoria

Definición (pLTS derivado)**.** *Sea $\mathcal{M} = \langle S, \Sigma, \delta, s_0 \rangle$ un MDP y $\pi \in Str_{\mathcal{M}}$ una estrategia sin memoria, entonces podemos derivar un pLTS $\mathcal{M}_\pi = \langle S_\pi, \Sigma_\pi, \delta_\pi, s_0^\pi \rangle$, donde:*

- $S_\pi = S$,
- $\Sigma_\pi = \Sigma$,
- $s_0^\pi = s_0$,
- $\delta_\pi(s, a, s') = \pi(s)(a) \cdot \delta(s, a)(s')$.

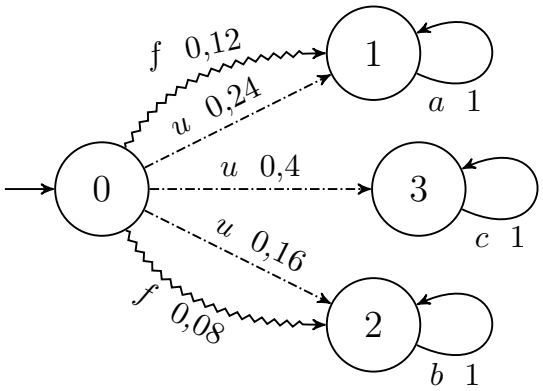


Figura 10: pLTS $\mathcal{M}_\pi$ derivado del MDP de la Figura 9.

Ejemplo: Volviendo al MDP de la Figura 9, supongamos que nuestra estrategia sin memoria es $\pi \in Str_{\mathcal{M}}$, donde

$$\pi = \begin{cases} 0 \mapsto [u \mapsto 0{,}8; f \mapsto 0{,}2] \\ 1 \mapsto [a \mapsto 1] \\ 2 \mapsto [b \mapsto 1] \\ 3 \mapsto [c \mapsto 1] \end{cases}$$

El pLTS $\mathcal{M}_\pi$ derivado a partir de la estrategia puede verse en la Figura 10.

4.1.5. Probabilidad de caminos y trazas en un MDP

En un pLTS $\mathcal{Q}$, dada una traza $w \in \mathcal{L}^*(\mathcal{Q})$, podemos calcular su probabilidad de ocurrencia sumando las probabilidades de todos los caminos que recorren dicha traza, obteniendo así una probabilidad entre 0 y 1. En cambio, si trabajamos con MDP, la probabilidad de cada camino no puede ser calculada hasta que no se defina una estrategia que resuelva el no determinismo del MDP, ya que dicha probabilidad depende de los valores asignados por la estrategia a cada evento disponible en cada estado del camino. Por lo tanto, si $\mathcal{M}$ es un MDP, entonces las probabilidades de ocurrencia de caminos y trazas en $\mathcal{M}$ se dejan expresadas en términos de los valores asignados por la estrategia, y se calculan de la siguiente manera:

- la *probabilidad de ocurrencia del camino* $\rho = s_0 \cdot a_1 \cdot s_1 \cdots a_n \cdot s_n \in Path^*_{\mathcal{M}}$

está dada por

$$Pr(\rho) = \prod_{i=0}^{n-1} \pi(s_i)(a_{i+1}) \cdot \delta(s_i, a_{i+1})(s_{i+1})$$

- la *probabilidad de ocurrencia de la traza* $w \in \mathcal{L}^*(\mathcal{M})$ está dada por

$$Pr(w) = \sum_{\{\rho \in Path^*_{\mathcal{M}} | tr(\rho) = w\}} Pr(\rho)$$

Ejemplo: Para calcular la probabilidad de la traza abc en el MDP de la Figura 11, hay que tener en consideración dos caminos distintos que siguen esta traza: $0a2b0c0$ y $0a1b4c4$

$$\begin{aligned} Pr(abc) &= Pr(0a2b0c0) + Pr(0a1b4c4) \\ &= \pi(0)(a) \cdot \overbrace{\delta(0,a)(2)}^{=0,7} \cdot \overbrace{\pi(2)(b)}^{=1} \cdot \overbrace{\delta(2,b)(0)}^{=1} \cdot \pi(0)(c) \cdot \overbrace{\delta(0,c)(0)}^{=1} \\ &+ \pi(0)(a) \cdot \underbrace{\delta(0,a)(1)}_{=0,3} \cdot \underbrace{\pi(1)(b)}_{=1} \cdot \underbrace{\delta(1,b)(4)}_{=0,6} \cdot \pi(4)(c) \cdot \underbrace{\delta(4,c)(4)}_{=1} \\ &= \pi(0)(a) \cdot 0{,}7 \cdot \pi(0)(c) + \pi(0)(a) \cdot 0{,}3 \cdot 0{,}6 \cdot \pi(4)(c) \end{aligned}$$

Como puede verse, la probabilidad de la traza abc se expresa en término de los valores que la estrategia debe determinar.

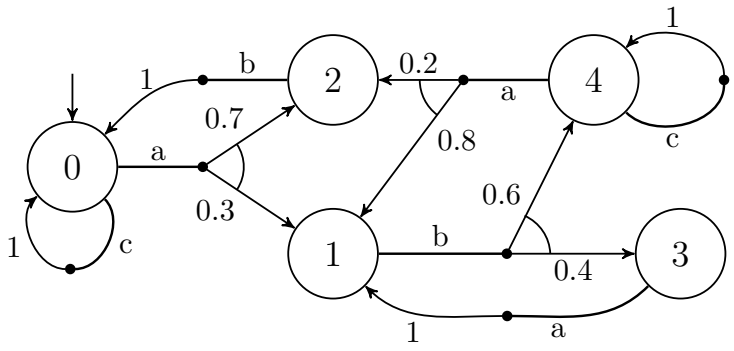


Figura 11

4.2. Diagnosticabilidad sobre MDP

La verificación de diagnosticabilidad sobre sistemas discretos consiste en determinar la existencia de un par de trazas indistinguibles observacionalmente en las cuales solamente una de ellas presenta una falla. Sobre pLTS planteamos un análisis sobre sistemas no diagnosticables mediante el cálculo de algunas probabilidades sobre la no diagnosticabilidad y la ocurrencia de fallas en el sistema. En MDP este mismo análisis no puede realizarse de manera similar, ya que el no determinismo del sistema no lo permite. Para lograr resultados parecidos a los que se plantean en el análisis sobre pLTS, nosotros proponemos un enfoque que consiste en tomar como referencia para el análisis al pLTS derivado de la estrategia que determiniza al MDP en el "peor caso posible", y a partir de éste determinar si es necesario modificar el sistema para reducir la probabilidad de ocurrencia de trazas problemáticas. Como dijimos en la Sección 3.2.6 cuando analizamos diagnosticabilidad sobre pLTS, consideramos al "peor caso posible" como aquel en el cual la probabilidad de ocurrencia de una traza no diagnosticable con una falla es idéntica a la probabilidad de ocurrencia de una traza no diagnosticable sin ninguna falla. Entonces dado un MDP, nuestro objetivo es calcular los valores de no diagnosticabilidad del peor pLTS que pueda derivarse mediante alguna estrategia. En esta sección presentamos nuestro principal aporte.

4.2.1. Diagnosticador de decisión

De manera similar al análisis de diagnosticabilidad realizado sobre pLTS, utilizamos un Diagnosticador construido a partir de la especificación del MDP, llamado **Diagnosticador de decisión**. Su construcción es similar a la del Diagnosticador estocástico, pero difiere solamente en los elementos de las matrices de transiciones entre estados, los cuales ahora contienen información sobre posibles elecciones no deterministas que una estrategia necesita resolver.

Definición (Diagnosticador de decisión)**.** *Un Diagnosticador de decisión asociado a un MDP* $\mathcal{M} = \langle S, \Sigma, \delta, s_0 \rangle$ *es una 5-úpla* $\mathcal{D} = \langle Q_d, \Sigma_d, \delta_d, q_0, \Phi \rangle$*, donde:*

- $Q_d \subseteq \mathcal{P}(S \times \{N, F\})$ *es el conjunto de estados,*
- Σ_d *es el conjunto de eventos,*

- $\delta_d : Q_d \times \Sigma_d \to Q_d$ *es la función de transición entre estados,*
- q_0 *es el estado inicial,*
- $\Phi : Q_d \times \Sigma_d \to [0,1]^{n \times m}$ *es la función de probabilidad de transiciones para los componentes de cada estado.*

El Diagnosticador de decisión se construye de manera similar al Diagnosticador estocástico de la Sección 3.2.2, salvo por la función Φ, cuya diferencia radica en que las probabilidades de los elementos de las matrices ahora pueden estar expresadas en base a decisiones que una estrategia debe resolver. De esta manera el Diagnosticador de decisión es un Diagnosticador estocástico paramétrico, cuyo parámetro es la estrategia que resuelve el no determinismo. Sean $(s_i, l_i) = comp_{q,i}$ y $(s_j, l_j) = comp_{\delta_d(q,a),j}$ para algún estado $q \in Q_d$ y $a \in \Sigma_d$, entonces

$$\Phi_{i,j}(q,a) = \sum_{\{w \in \Sigma_u^* a \,|\, s_i \overset{w}{\Rightarrow} s_j \ \wedge \ l_j = LP(w,l_i)\}} Pr(w)$$

donde $Pr(w)$ es la función de probabilidad de ocurrencia de trazas en MDP definida en la Sección 4.1.5 y LP es la función definida en la Sección 3.2.2

Ejemplo: A partir del MDP de la Figura 9, podemos construir el Diagnosticador de decisión de la Figura 12, en el cual podemos ver que las matrices de transiciones están formadas por probabilidades que dependen de la elección de una estrategia.

4.2.2. Utilidad del Diagnosticador de decisión

A diferencia del Diagnosticador estocástico, el Diagnosticador de decisión tiene una utilidad limitada a la hora de realizar diagnosis del sistema. Esto se debe a que el no determinismo presente en MDP no permite estimar la probabilidad de estar en cada una de las componentes alcanzadas por una traza observable de eventos. La diagnosis en MDP se limita entonces a seguir los estados alcanzados por cada observación en el sistema, donde cada estado contiene las componentes con los posibles estados actuales del sistema y una etiqueta que marca si ha ocurrido una falla en el camino.

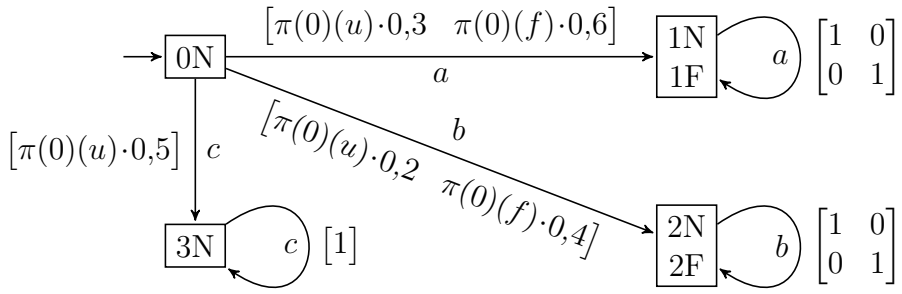


Figura 12: Diagnosticador asociado al MDP de la Figura 9

Debido a las limitaciones para realizar diagnosis, la principal utilidad del Diagnosticador de decisión es para hacer un análisis de la diagnosticabilidad del sistema, calculando valores sobre su diagnosticabilidad considerando la estrategia que resulte más desfavorable. Este análisis es bastante similar al análisis realizado sobre el Diagnosticador estocástico en la Sección 3.2.3, pero a diferencia de aquel análisis, aquí se procede transformando el problema en un problema de optimización que concluye con la elección de una de las estrategias más desfavorables para el MDP analizado.

4.2.3. Diagnosticabilidad usando el Diagnosticador de decisión

Analizar si un MDP es o no diagnosticable se reduce nuevamente a verificar si el LTS obtenido de quitar las probabilidades del MDP cumple con la propiedad de diagnosticabilidad:

$$\forall \sigma_1, \sigma_2 \in \mathcal{L}^{\omega}(\mathcal{Q}) : obs(\sigma_1) = obs(\sigma_2), \text{ si } f \in \sigma_1 \text{ entonces } f \in \sigma_2$$

Esta verificación puede llevarse a cabo mediante el método "Twin-Plant" de la Sección 2.2.1. Si tras aplicar este método llegamos a la conclusión de que el sistema es no diagnosticable, entonces nos gustaría cuantificar el grado de no diagnosticabilidad del sistema, tal como lo hicimos en la Sección 3.2.3, donde analizamos los valores para $p_{\neg d}$, p_F y $p_{F|\neg d}$, entre otros, para determinar si es o no necesario introducir cambios en el sistema que mejoren los resultados obtenidos. Como dijimos anteriormente, estos valores no pueden ser calculados directamente a partir de un MDP debido al no determinismo presente en estos modelos. Suponiendo que este no determinismo se debe

a acciones que no pueden ser controladas, quisiéramos analizar el caso que arroja los resultados más desfavorables en términos de diagnosticabilidad del sistema. En la Sección 3.2.6 comentamos que el peor caso posible se da cuando existe el mayor nivel de incertidumbre con respecto a la ocurrencia de una falla al momento de observar una traza no diagnosticable, es decir, cuando el resultado de $\left|p_{F|\neg d} - p_{\neg F|\neg d}\right|$ es igual o muy cercano a 0.

El análisis de diagnosticabilidad sobre MDP consiste entonces en analizar qué tan malo puede llegar a ser un pLTS derivado de alguna estrategia aplicada al MDP.

Para obtener la estrategia que derive un pLTS a partir del MDP, obtenemos ecuaciones para los valores de $p_{\neg d}$, p_F, $p_{F|\neg d}$, etc. a partir de la cadena de Markov extraída del Diagnosticador de decisión, y luego minimizamos el valor de $\left|p_{F|\neg d} - p_{\neg F|\neg d}\right|$ mediante técnicas de optimización.

4.2.4. Extracción de la cadena de Markov

Para comenzar con el cálculo de las ecuaciones para los valores de diagnosticabilidad, es necesario extraer la cadena de Markov asociada al conjunto de todas las transiciones del Diagnosticador. Esta cadena nos permite hacer un análisis del comportamiento asintótico de las trazas aceptadas por el sistema, y para obtenerla utilizamos la función $\Omega : Q_d \times Q_d \to [0,1]^{n\times m}$ definida de la misma manera que en la Sección 3.2.3:

$$\Omega(q_i, q_j) = \sum_{\{a\in\Sigma_d | \delta_d(q_i,a)=q_j\}} \Phi(q_i, a)$$

Entonces, si $\mathcal{M} = \langle S, \Sigma, \delta, s_0\rangle$ es un MDP y $\mathcal{D} = \langle Q_d, \Sigma_d, \delta_d, q_0, \Phi\rangle$ es el Diagnosticador de decisión asociado a $\mathcal{M}$, la cadena de Markov asociada a $\mathcal{D}$ es $\Pi(\mathcal{D}) = \langle S^\Pi, \delta^\Pi, s_0^\Pi\rangle$, donde

- $S^\Pi \subseteq Q_d \times S \times \{N, F\}$ es el conjunto de todas las componentes de $\mathcal{D}$, donde si $(q, s, l) \in S^\Pi$ entonces $(s, l) \in q$
- δ^Π está dada por la siguiente matriz de transiciones:

$$tr = \begin{bmatrix} \Omega(q_0, q_0) & \cdots & \Omega(q_0, q_n) \\ \vdots & \ddots & \vdots \\ \Omega(q_n, q_0) & \cdots & \Omega(q_n, q_n) \end{bmatrix}$$

- $s_0^\Pi = (q_0, s_0, N)$ es el estado inicial

Ejemplo: A partir del Diagnosticador de decisión de la Figura 12 calculado a partir del MDP de la Figura 9, podemos extraer un DTMC $\Pi(\mathcal{D})$ cuyos estados son $\{(q_0, 0, N), (q_1, 1, N), (q_1, 1, F), (q_2, 2, N), (q_2, 2, F), (q_3, 3, N)\}$, el estado inicial es $(q_0, 0, N)$ y cuyas transiciones entre estados están dadas por la siguiente matriz:

$$tr = \begin{bmatrix} 0 & \pi(0)(u)\cdot 0{,}3 & \pi(0)(f)\cdot 0{,}6 & \pi(0)(u)\cdot 0{,}2 & \pi(0)(f)\cdot 0{,}4 & \pi(0)(u)\cdot 0{,}5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

donde cada valor se corresponde con la probabilidad de ir de un estado a otro, siguiendo el orden en el que los estados fueron listados.

4.2.5. Comportamiento asintótico de la cadena de Markov

En el caso de pLTS, el análisis del comportamiento asintótico de la cadena de Markov extraída del Diagnosticador estocástico permitía extraer valores sobre la diagnosticabilidad del pLTS. A diferencia de aquel análisis, cuando analizamos la cadena de Markov extraída de un Diagnosticador de decisión nos encontramos con que sus elementos están expresados en términos de una estrategia que resuelve las elecciones no deterministas. Esta diferencia hace que el análisis asintótico de la cadena de Markov para el cálculo de la diagnosticabilidad de un MDP no nos otorgue valores concretos, como sucede en el caso de los pLTS. Estos valores ahora son paramétricos, cuyo parámetro es la estrategia que resuelva la decisiones no deterministas.

Para obtener las ecuaciones para los valores $p_{\neg d}$, p_F, $p_{F|\neg d}$, $p_{\neg F|\neg d}$ y avg_{step}, introducidos en la Sección 3.2.3, procedemos de manera similar a como hicimos en la Sección 3.2.4, es decir:

1. Clasificamos los estados de $\Pi(\mathcal{D})$: extraemos el conjunto de estados recurrentes $\{\lambda_1, \ldots, \lambda_k\}$ agrupados en clases de recurrencia $\{\zeta_1, \ldots, \zeta_h\}$, y el conjunto de estados transitorios $\{\nu_1, \ldots, \nu_r\}$. Para hacer esta clasificación necesitamos conocer bien las probabilidades de transiciones entre estados. La cadena de Markov extraída de un Diagnosticador de decisión puede no tener valores concretos en algunas de sus transiciones, y dependiendo de los valores que asigne una estrategia a distintos

eventos, la clasificación de estados realizada en este paso puede variar drásticamente. Para evitar problemas asumimos que las transiciones del modelo cumplen con un requisito de *fairness*, con lo cual las estrategias que resuelven el no determinismo sobre un MDP nunca deben truncar transiciones asignándoles una probabilidad nula.

2. Reorganizamos las filas y columnas de la matriz de transiciones de $\Pi(\mathcal{D})$ en su *forma canónica*, obteniendo así una matriz de transiciones de la forma:

$$tr = \begin{bmatrix} Q & R \\ 0 & C \end{bmatrix}, \qquad C = \begin{bmatrix} tr_{\zeta_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & tr_{\zeta_h} \end{bmatrix}$$

   donde tr_{ζ_i} es la matriz de transiciones de la clase de recurrencia ζ_i.

3. Computamos la matriz fundamental de la cadena de Markov, dada por $N = (I - Q)^{-1}$, y la matriz de absorción, dada por $B = N \cdot R$.

4. Sea $\zeta_{\neg d}$ el subconjunto de clases de recurrencia que contienen solo estados asociados a componentes ambiguos del Diagnosticador, y sea ζ_F el subconjunto de clases de recurrencia que contienen solo estados asociados a componentes marcados con F en el Diagnosticador, entonces con ayuda de las matrices N y B podemos calcular los siguientes valores:

   - $p_{\neg d}$ es la probabilidad de ser absorbido en una de las clases de $\zeta_{\neg d}$ a partir del estado inicial. Está dada por

$$p_{\neg d} = \sum_{\zeta \in \zeta_{\neg d}} \sum_{\lambda \in \zeta} B_{s_0^{\Pi}, \lambda}$$

   - p_F es la probabilidad de ser absorbido en una de las clases de ζ_F a partir del estado inicial. Está dada por

$$p_F = \sum_{\zeta \in \zeta_F} \sum_{\lambda \in \zeta} B_{s_0^{\Pi}, \lambda}$$

- $p_{F\wedge\neg d}$ es la probabilidad de ser absorbido en una de las clases pertenecientes tanto a $\zeta_{\neg d}$ como a ζ_F a partir del estado inicial. Usando la fórmula de Bayes calculamos el valor de $p_{F|\neg d}$ como

$$p_{F|\neg d} = \frac{p_{F\wedge\neg d}}{p_{\neg d}} = \frac{\sum_{\zeta\in(\zeta_{\neg d}\cap\zeta_F)} \sum_{\lambda\in\zeta} B_{s_0^{\Pi},\lambda}}{\sum_{\zeta\in\zeta_{\neg d}} \sum_{\lambda\in\zeta} B_{s_0^{\Pi},\lambda}}$$

- $p_{\neg F|\neg d}$ se obtiene de manera similar, sólo que ahora nos interesan las clases pertenecientes a $\zeta_{\neg d}$ pero no a ζ_F, es decir

$$p_{\neg F|\neg d} = \frac{\sum_{\zeta\in(\zeta_{\neg d}-\zeta_F)} \sum_{\lambda\in\zeta} B_{s_0^{\Pi},\lambda}}{\sum_{\zeta\in\zeta_{\neg d}} \sum_{\lambda\in\zeta} B_{s_0^{\Pi},\lambda}} = 1 - p_{F|\neg d}$$

- avg_{step} es la cantidad promedio de pasos necesarias para llegar a alguna clase de recurrencia a partir del estado inicial. Está dada por

$$avg_{step} = \sum_{i=1}^{r} N_{s_0^{\Pi},\nu_i}$$

Ejemplo: En el ejemplo anterior, la matriz de transiciones ya está en forma canónica:
$R = \big[\pi(0)(u)\cdot 0{,}3 \quad \pi(0)(f)\cdot 0{,}6 \quad \pi(0)(u)\cdot 0{,}2 \quad \pi(0)(f)\cdot 0{,}4 \quad \pi(0)(u)\cdot 0{,}5\big]$,
$Q = [0]$ y $C = I$.
Por lo tanto $N = (I - Q)^{-1} = I$ y $B = N\cdot R = I\cdot R = R$. Las clases de recurrencia con componentes ambiguas son $\{(q_1, 1, N)\}$, $\{(q_1, 1, F)\}$, $\{(q_2, 2, N)\}$ y $\{(q_2, 2, F)\}$, mientras que las clases de recurrencia etiquetadas con fallas son $\{(q_1, 1, F)\}$ y $\{(q_2, 2, F)\}$. Usando esta información obtenemos las siguientes ecuaciones, cuyos valores dependen únicamente de la asignación de valores a $\pi(0)(u)$ y $\pi(0)(f)$:

$$
\begin{aligned}
p_{\neg d} &= 0{,}5 \cdot \pi(0)(u) + \pi(0)(f) \\
p_F &= \pi(0)(f) \\
p_{F|\neg d} &= \frac{\pi(0)(f)}{0{,}5 \cdot \pi(0)(u) + \pi(0)(f)} \\
p_{\neg F|\neg d} &= \frac{0{,}5 \cdot \pi(0)(u)}{0{,}5 \cdot \pi(0)(u) + \pi(0)(f)} \\
avg_{step} &= 1
\end{aligned}
$$

A partir de estas ecuaciones, solo resta obtener una estrategia que minimice el valor de $\left|p_{F|\neg d} - p_{\neg F|\neg d}\right|$. Para lograrlo, formulamos este problema como un problema de optimización.

4.2.6. Problema de optimización

Un *problema de optimización* [3] consiste en encontrar la mejor solución dentro de un conjunto posible de soluciones que satisfacen ciertas restricciones para un problema. La mejor solución posible es aquella que maximiza o minimiza el valor de una función. La forma estándar de un problema de optimización es la siguiente:

$$
\begin{aligned}
\underset{X}{\text{minimizar}} \quad & f(X) \\
\text{sujeto a} \quad & g_i(X) \leq b_i, \quad i = 1, \ldots, m \\
& h_i(X) = c_i, \quad i = 1, \ldots, p
\end{aligned}
$$

donde:

- la función $f : \mathbb{R}^n \to \mathbb{R}$ es la *función objetivo* a ser minimizada,
- $g_i(X) \leq b_i$ son *restricciones de inigualdad* sobre X,
- $h_i(X) = c_i$ son *restricciones de igualdad* sobre X.

La definición estándar describe un problema de minimización. Sin embargo, un problema de maximización se consigue negando la función objetivo.

Los problemas de optimización son categorizados dependiendo de la forma y de la complejidad que presenten la función objetivo y las funciones de

restricciones, y de acuerdo al tipo de problema se aplican diferentes algoritmos que resuelven de manera eficiente cada uno de ellos. Dado que nuestro problema genera siempre restricciones lineales, en esta sección nos enfocamos solamente en la resolución de problemas con restricciones de este tipo. En el Apéndice A se explica de manera resumida como se resuelven estos problemas.

En resumen, para describir un problema de optimización debemos:

1. definir el conjunto de variables del problema,
2. dar restricciones sobre los valores que pueden tomar estas variables,
3. expresar la función objetivo en base a estas variables.

Como dijimos en la sección anterior, queremos obtener una estrategia que minimice el valor de $\left|p_{F|\neg d} - p_{\neg F|\neg d}\right|$. Éste es un problema de optimización en el cual:

1. las variables del problema están dadas por los valores que la estrategia asigna a cada evento disponible en cada estado del MDP, es decir, sea $\mathcal{M} = \langle S, \Sigma, \delta, s_0 \rangle$ un MDP, podemos representar a una estrategia $\pi \in Str_{\mathcal{M}}$ como un conjunto de variables $X = \{\pi(s)(a) \mid s \in S, a \in A(s)\}$
2. las restricciones sobre los valores de las variables están dadas por el hecho de tratarse de probabilidades, es decir,
   - $\forall s \in S, \forall a \in A(s), 0 \leq \pi(s)(a) \leq 1$ (restricciones de inigualdad)[2]
   - $\forall s \in S, \sum_{a \in A(s)} \pi(s)(a) = 1$ (restricciones de igualdad)

   las cuales son restricciones lineales sobre las variables en X y definen regiones acotadas donde buscar posibles soluciones,
3. la función objetivo a minimizar es $f(X) = \left|p_{F|\neg d} - p_{\neg F|\neg d}\right|$, la cual hace uso de decisiones tomadas por la estrategia, es decir, de variables en X.

[2]Incluímos el valor 0 a pesar de que en la Sección 4.2.5 requerimos no truncar transiciones del modelo debido a que los algoritmos de resolución de problemas de optimización no admiten la utilización de los operadores “>” y “<”. Soluciones que utilicen el valor 0 representarán en verdad valores tan cercanos a 0 como se requiera.

Resolver este problema de optimización resulta en una estrategia π^*, llamada **estrategia óptima**, puesto que optimiza el valor de la función objetivo. Aplicar esta estrategia al MDP $\mathcal{M}$ resulta en un pLTS $\mathcal{M}_{\pi^*}$ que representa el comportamiento más desfavorable en términos de diagnosticabilidad que puede tomar el sistema.

Ejemplo: Para obtener la estrategia óptima para el MDP de nuestro ejemplo, expresamos el problema como un problema de optimización:

1. El conjunto de variables es $X = \{\pi(0)(u),\ \pi(0)(f),\ \pi(1)(a),\ \pi(2)(b),\ \pi(3)(c)\}$
2. Las restricciones son:
   - $0 \leq \pi(0)(u) \leq 1$
   - $0 \leq \pi(0)(f) \leq 1$
   - $0 \leq \pi(1)(a) \leq 1$
   - $0 \leq \pi(2)(b) \leq 1$
   - $0 \leq \pi(3)(c) \leq 1$
   - $\pi(0)(u) + \pi(0)(f) = 1$
   - $\pi(1)(a) = 1$
   - $\pi(2)(b) = 1$
   - $\pi(3)(c) = 1$
3. la función objetivo a minimizar es

$$\begin{aligned} f(X) &= \left|p_{F\,|\,\neg d} - p_{\neg F\,|\,\neg d}\right| \\ &= \left|\frac{\pi(0)(f)}{0{,}5 \cdot \pi(0)(u) + \pi(0)(f)} - \frac{0{,}5 \cdot \pi(0)(u)}{0{,}5 \cdot \pi(0)(u) + \pi(0)(f)}\right| \\ &= \frac{|\pi(0)(f) - 0{,}5 \cdot \pi(0)(u)|}{0{,}5 \cdot \pi(0)(u) + \pi(0)(f)} \end{aligned}$$

Al optimizar el valor de la función objetivo obtenemos una asignación a las variables de X que se corresponde con la siguiente estrategia óptima:

$$\pi^* = \begin{cases} 0 \mapsto [u \mapsto \frac{2}{3}; f \mapsto \frac{1}{3}] \\ 1 \mapsto [a \mapsto 1] \\ 2 \mapsto [b \mapsto 1] \\ 3 \mapsto [c \mapsto 1] \end{cases}$$

Esta estrategia lleva el valor de $\left|p_{F\,|\,\neg d} - p_{\neg F\,|\,\neg d}\right|$ a 0, es decir, la estrategia aplicada al MDP genera un pLTS en el cual la probabilidad de ocurrencia de

fallas en trazas no diagnosticables es idéntica a la probabilidad de ausencia de fallas en trazas no diagnosticables. Aplicando esta estrategia obtenemos los siguientes resultados:

- $p_{\neg d} = \frac{2}{3}$
- $p_{F|\neg d} = \frac{1}{2}$
- $avg_{step} = 1$
- $p_F = \frac{1}{3}$
- $p_{\neg F|\neg d} = \frac{1}{2}$

A partir de estos valores concluimos que el sistema es altamente no diagnosticable y su observabilidad debería ser incrementada.

4.2.7. Observaciones finales sobre el algoritmo

1. El algoritmo presentado deriva una estrategia sin memoria, ya que a partir de este tipo de estrategias se obtiene directamente un pLTS con una estructura similar a la del MDP. Consideramos únicamente estrategias sin memoria porque creemos que todo otro tipo de estrategia puede reducirse a una sin memoria cuyos resultados sean similares.

   Un esquema de cómo reducir una estrategia con memoria a una estrategia sin memoria sería el siguiente:

   a) Dado un MDP $\mathcal{M} = \langle S, \Sigma, \delta, s_0 \rangle$, supongamos que tenemos una estrategia con memoria $\pi_1 \in Str_{\mathcal{M}}$ que asigna a elementos $\rho \in Path^*_{\mathcal{M}}$ una distribución de probabilidad sobre $A(last(\rho))$.

   b) La estrategia sin memoria $\pi_2 \in Str_{\mathcal{M}}$ que emula el comportamiento de π_1 se define de la siguiente manera: Sea $s \in S$, $\pi_2(s)$ es la distribución de probabilidad que resulta de promediar cada distribución asignada por la estrategia π_1 a todos los caminos que terminen en el estado s.

2. La estrategia obtenida mediante este algoritmo es aquella que representa el peor comportamiento que puede darse en un MDP respetando su estructura original. Este algoritmo no contempla comportamientos que consisten en truncar transiciones del modelo, pues cada una de estos truncamientos puede resultar en un Diagnosticador diferente del modelo, y analizar por separado cada uno de ellos implica un incremento exponencial en la complejidad del método.

3. Considerar a la estrategia óptima como aquella que minimice el valor de $\left|p_{F\,|\,\neg d} - p_{\neg F\,|\,\neg d}\right|$ es subjetivo, y basta con cambiar la función objetivo

en el problema de optimización para considerar otras posibles interpretaciones del caso más desfavorable con respecto a la diagnosticabilidad del sistema (por ejemplo, la estrategia que minimice $p_{\neg d}$). Si se considera que las acciones del sistema son controlables, se puede derivar una estrategia que trate de evitar la ocurrencia de fallas al minimizar el valor de p_F.

5. Análisis de Complejidad

En esta sección repasamos los algoritmos utilizados para el análisis de diagnosticabilidad y analizamos brevemente sus complejidades.

5.1. Algoritmo presentado para LTS

El algoritmo que utilizamos para la verificación de diagnosticabilidad sobre LTS es el "Twin-Plant", introducido en la Sección 2.2.1. Dado un LTS $\mathcal{N} = \langle S, \Sigma, \delta, s_0 \rangle$, este método consiste en:

1. construir el LTS $\mathcal{N}_o = \langle S_o, \Sigma_o, \delta_o, s_0^o \rangle$,
2. computar $\mathcal{N}_d = \mathcal{N}_o \parallel \mathcal{N}_o$,
3. verificar si en $\mathcal{N}_d$ (o en su versión optimizada) se da que $s \overset{w}{\Rightarrow} s$, para $w \neq \epsilon$ y para algún estado $s = ((s_1, l_1), (s_2, l_2))$ tal que $l_1 \neq l_2$.

A partir del algoritmo, podemos saber que el número de estados en $\mathcal{N}_o$ es a lo sumo $2\,|S|$ y el número de transiciones en $\mathcal{N}_o$ es a lo sumo $(2\,|S|)^2 \times |\Sigma_o|$. Como $\mathcal{N}_d = \mathcal{N}_o \parallel \mathcal{N}_o$, el número de estados en $\mathcal{N}_d$ es a lo sumo $(2\,|S|)^2$, y el número de transiciones en $\mathcal{N}_d$ es a lo sumo $(2\,|S|)^4 \times |\Sigma_o|$.

La complejidad de ejecutar el primer paso del algoritmo, en el cual se construye $\mathcal{N}_o$, es $\mathcal{O}(|S|^2 \times |\Sigma_o|)$, mientras que la del segundo paso, en el cual $\mathcal{N}_d$ es construido, es $\mathcal{O}(|S|^4 \times |\Sigma_o|)$. La complejidad del tercer paso del algoritmo, el cual detecta la presencia de ciertos ciclos en $\mathcal{N}_d$, es lineal en el número de estados y transiciones, es decir, su complejidad es $\mathcal{O}(|S|^4 \times |\Sigma_o|)$. Notar que las etiquetas de las transiciones son irrelevantes al momento de detectar la presencia de ciclos. Por lo tanto, la complejidad del algoritmo es polinomial en el número de estados y lineal en la cantidad de eventos en $\mathcal{N}$.

5.2. Algoritmo presentado para pLTS

El algoritmo presentado para el análisis de diagnosticabilidad se basa en la construcción de un Diagnosticador, el cual permite un análisis estático y dinámico sobre la ocurrencia de fallas en el sistema. Dado un pLTS $\mathcal{Q} = \langle S, \Sigma, \delta, s_0 \rangle$, el algoritmo consiste en:

1. construir el Diagnosticador estocástico $\mathcal{D} = \langle Q_d, \Sigma_d, \delta_d, q_0, \Phi \rangle$,

2. extraer la cadena de Markov $\Pi(\mathcal{D}) = \langle S^{\Pi}, \delta^{\Pi}, s_0^{\Pi} \rangle$,

3. clasificar los estados de $\Pi(\mathcal{D})$,

4. computar la matriz fundamental y la matriz de absorción.

Al construir el Diagnosticador, el número de estados es a lo sumo $2^{2|S|}$, pues $Q_d \subseteq \mathcal{P}(S \times \{N, F\})$, y las transiciones del Diagnosticador son a lo sumo $2^{2|S|} \times |\Sigma_o|$.

Cuando extraemos la cadena de Markov, obtenemos una cadena con un estado por cada componente del Diagnosticador ($S^{\Pi} \subseteq Q_d \times S \times \{N, F\}$), es decir que a lo sumo la cadena tiene $2^{2|S|} \times |S| \times 2$ estados, y es necesario definir a lo sumo $(2^{2|S|} \times |S| \times 2)^2$ transiciones. Luego es necesario clasificar los estados de la cadena, lo cual es lineal en la cantidad de estados y transiciones de la cadena. Finalmente es necesario calcular la matriz fundamental y la de absorción para obtener los valores de diagnosticabilidad del sistema. Este último paso requiere la inversión de una matriz, lo cual tiene una complejidad cúbica aplicando eliminación de Gauss–Jordan (aunque existen también algoritmos cuya complejidad es polinomial de grado $\approx 2,373$).

Por lo tanto, la complejidad de ejecutar el primer paso del algoritmo, en el cual se construye $\mathcal{D}$, es $\mathcal{O}(2^{2|S|} \times |\Sigma_o|)$, la complejidad de ejecutar el segundo paso, en el cual se obtiene $\Pi(\mathcal{D})$, es $\mathcal{O}(2^{4|S|})$ y la complejidad de ejecutar los dos últimos pasos del algoritmo, a partir de los cuales obtenemos los valores de diagnosticabilidad, es aproximadamente $\mathcal{O}(2^{6|S|})$. Entonces, la complejidad total del algoritmo es $\mathcal{O}(2^{2|S|} \times |\Sigma_o| + 2^{6|S|})$, es decir, exponencial en la cantidad de estados.

5.3. Algoritmo presentado para MDP

El algoritmo utilizado para MDP es similar al utilizado para pLTS, agregando un nuevo paso que consiste en resolver el problema de optimización. Dado un MDP $\mathcal{M} = \langle S, \Sigma, \delta, s_0 \rangle$, el algoritmo consiste en:

1. construir el Diagnosticador de decisión $\mathcal{D} = \langle Q_d, \Sigma_d, \delta_d, q_0, \Phi \rangle$,

2. extraer la cadena de Markov $\Pi(\mathcal{D}) = \langle S^{\Pi}, \delta^{\Pi}, s_0^{\Pi} \rangle$,

3. clasificar los estados de $\Pi(\mathcal{D})$,

4. computar la matriz fundamental y la matriz de absorción,

5. resolver el problema de optimización.

El análisis de la complejidad de los primeros cuatro pasos no presenta diferencias con respecto al algoritmo utilizado sobre pLTS. El problema de optimización se compone a lo sumo de $|S| \times |\Sigma|$ variables, cuyos valores se ven acotados por restricciones que obligan a cada variable a tomar un valor entre 0 y 1 y que la suma de los valores de las variables que se correspondan con cada estado sumen 1.

A diferencia de los algoritmos utilizados para resolver problemas de optimización lineal, los algoritmos utilizados para resolver problemas de optimización no lineales son raramente analizados en términos de complejidad. Los principales métodos utilizados para la resolución de estos problemas son numéricos e iterativos, en los cuales el análisis de complejidad es sustituido por un análisis sobre el grado de convergencia hacia la solución óptima [9].

En nuestro caso, la complejidad de la función a minimizar en el análisis de diagnosticabilidad varia dependiendo del modelo. Por esto, la estructura de dicha función puede ser de muchas maneras: constante, lineal fraccional (en los mejores casos), y hasta compuesta por un gran número de variables multiplicadas entre si.

6. Ejemplo ilustrativo

En esta sección presentamos un ejemplo ilustrativo en el cual aplicamos el algoritmo para el análisis de diagnosticabilidad en sistemas modelados como MDP.

6.1. Presentación

Una fábrica se dedica a la elaboración de productos derivados de la manzana. Tiene en stock una gran cantidad de las mismas y las utiliza como materia prima en la elaboración de dos productos: sidra y jugo de manzana. Para la elaboración de una sidra de excelente calidad, las manzanas utilizadas deben cumplir ciertos niveles de calidad (correcto tamaño, maduración justa, niveles de acidez adecuados, etc.), mientras que para la elaboración de jugo no es necesario que las manzanas cumplan con requisitos tan estrictos, es decir, se puede utilizar cualquier manzana para elaborar jugo, pero para la elaboración de sidra las manzanas deben ser correctamente seleccionadas.

Por lo tanto, la fábrica necesita clasificar las manzanas como aptas para la elaboración de sidra o para la elaboración de jugo dependiendo de si cumplen o no con dichas características. El proceso de clasificación se automatiza totalmente mediante el uso de una gran cantidad de máquinas que clasifican individualmente cada una de las manzanas realizando una serie de pruebas sobre ellas.

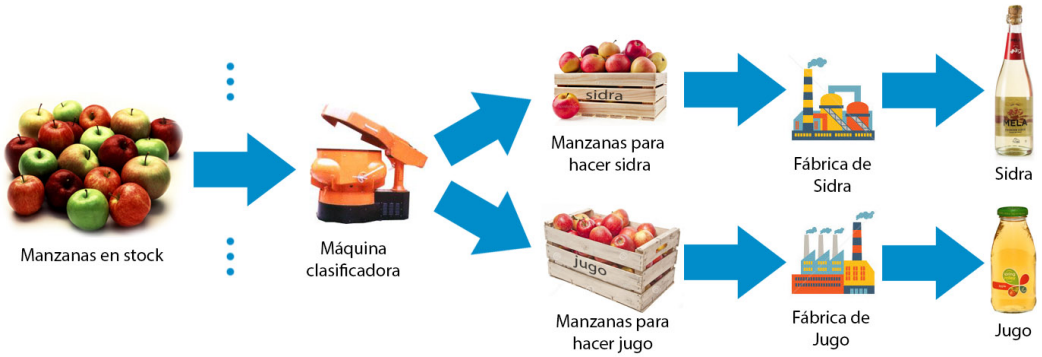


Figura 13: Esquema del proceso de elaboración de los productos

6.2. Máquinas clasificadoras

Las máquinas clasificadoras son las encargadas de decidir mediante una serie de pruebas si una manzana debe ser usada para la elaboración de sidra o para la elaboración de jugo. El funcionamiento de cada una de las máquinas es el siguiente:

1. Inicialmente la máquina informa que está preparada para el análisis de una manzana. En ese momento una manzana ingresa para su clasificación. Debido a que este proceso es mecánico, puede suceder que la manzana quede incorrectamente colocada para su análisis. En este caso, al momento de realizar las pruebas para la clasificación nunca se llegará a un veredicto, y la máquina continuará realizando pruebas hasta que en algún momento decida reiniciar todo el proceso para intentar que la manzana quede colocada correctamente.

2. Cada máquina es capaz de realizar pruebas de dos tipos diferentes para determinar el tipo de manzana. Estas pruebas son las siguientes:

   `A:` Se analiza un pequeña muestra extraída desde el interior de la manzana. El resultado de una prueba de este tipo es más certero, pero también más costoso. Si la muestra extraída es lo suficientemente buena, la manzana se clasifica correctamente. Sin embargo, con una baja probabilidad, puede suceder que la muestra extraída no sea lo suficientemente buena para lograr un resultado, por lo que haría falta realizar otra prueba o incluso, si la muestra extraída no refleja correctamente la composición general del resto de la manzana, el resultado puede concluir en una clasificación errónea.

   `B:` Se analiza superficialmente una sección de la cáscara de la manzana. Al ser una prueba menos invasiva, el resultado de esta prueba es menos certero que la anterior. Al realizar esta prueba pueden darse tres posibilidades: que la sección analizada permita realizar una correcta clasificación; que se analice una sección que no permita distinguir el tipo de manzana analizada y sea necesario repetir la prueba; o que se concluya una clasificación errónea por analizar una sección "engañosa" de la manzana.

3. Finalmente, cuando se decide el tipo de la manzana analizada, se lo notifica con una señal y luego la máquina queda en un estado de sus-

pensión. Una vez que sus resultados son recogidos y la manzana es separada de acuerdo a su tipo, se vuelve a iniciar el proceso.

Para comprender mejor el funcionamiento de las máquinas clasificadoras, en la Figura 14 la modelamos como un MDP.

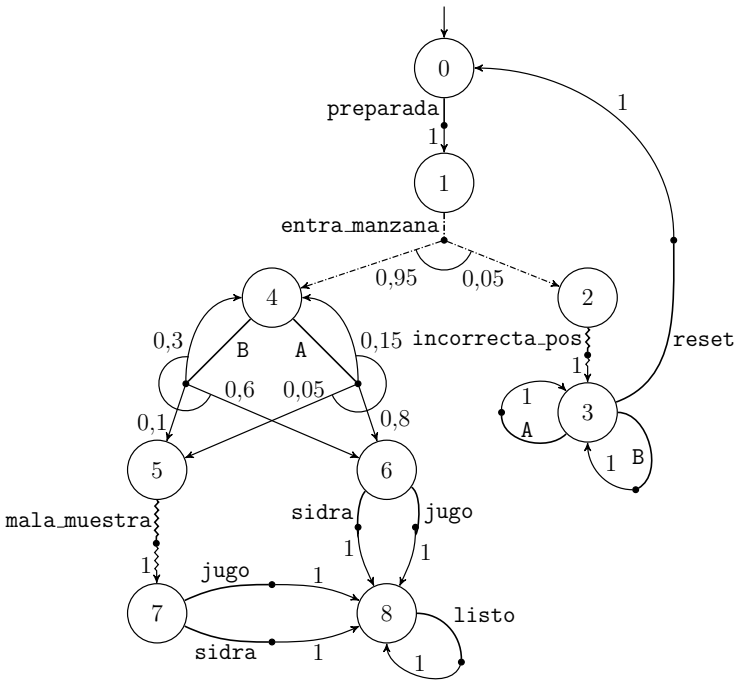


Figura 14: Modelo de una máquina clasificadora

El modelo se compone de ocho estados y siete eventos. Cuando la máquina está lista para iniciar el análisis de una manzana, emite la señal `preparada` y espera el ingreso de una manzana. Con `entra_manzana` se modela el ingreso de una manzana a la máquina, lo cual consiste en la correcta colocación de la manzana para que la máquina pueda realizar la clasificación. Como mencionamos previamente, debido a que este es un proceso mecánico, existe la chance de que la manzana no ingrese a la máquina o quede mal colocada, modelado con el evento `incorrecta_pos`, lo cual significa que la máquina intentará realizar repetidas pruebas en vano hasta que decida reiniciar el proceso con una señal `reset`. Cuando la manzana ingrese correctamente, la máquina decide si realizar una prueba `A` o `B` para clasificar la manzana. Cada

una de estas pruebas tienen probabilidades de conseguir clasificar correctamente la manzana (transiciones hacia el estado 6), clasificarla incorrectamente debido a la toma de una mala muestra (transiciones hacia el estado 5) o no conseguir información suficiente como para arriesgar un resultado, por lo que se intenta realizar otra prueba (transiciones que regresan al estado 4). El estado 6 representa la situación de haber logrado clasificar correctamente la manzana analizada, y a partir de este estado se notifica con una señal `sidra` o `jugo` la decisión tomada, para luego entrar en un estado de suspensión emitiendo constantemente señales `listo`. Por el contrario, en el estado 5 se modela la situación en la que se realiza una clasificación incorrecta con el evento `mala_muestra`, lo cual hace que en el estado 7 se notifique la decisión tomada con una señal `sidra` o `jugo` sin estar en lo correcto.

Un controlador central consulta periódicamente los *logs* de cada una de las máquinas para conocer y separar las manzanas analizadas a partir de las señales `sidra` o `jugo` emitidas. Una vez extraída la manzana de una máquina, ésta se reinicia para pasar al análisis de la siguiente manzana.

6.3. Análisis de Diagnosticabilidad del modelo

Si analizamos las trazas aceptadas por el modelo podemos ver que estamos ante un sistema no diagnosticable, ya que no se pueden distinguir, por ejemplo, las trazas que tomaron una mala muestra de otras que no lo hicieron.

A partir del modelo del problema se quieren averiguar posibles comportamientos de las máquinas clasificadoras:

- ¿Es posible que una máquina clasificadora se comporte de manera tal que tenga la misma probabilidad de fallar que de no hacerlo?
- ¿De qué manera debería comportarse una máquina para obtener la mayor y la menor probabilidad de ocurrencia de fallas?
- ¿Cómo sería el comportamiento de una máquina que distribuye uniformemente sus decisiones?

En las secciones siguientes haremos un análisis sobre los valores de diagnosticabilidad sobre el modelo de las máquinas clasificadoras para obtener estas respuestas.

Para la resolución de estas preguntas procedemos a aplicar el algoritmo presentado en la Sección 4 para hacer un análisis sobre los valores de diagnosticabilidad del modelo de las máquinas clasificadoras.

Por simpleza, representamos a cada evento con su primera letra, es decir que a partir de ahora el conjunto $\Sigma = \{$`preparada`, `entra_manzana`, `incorrecta_pos`, `A`, `B`, `reset`, `mala_muestra`, `sidra`, `jugo`, `listo`$\}$ pasa a ser $\Sigma = \{$`p`, `e`, `i`, `A`, `B`, `r`, `m`, `s`, `j`, `l`$\}$. Además, utilizamos π_s^e para denotar $\pi(s)(e)$.

6.3.1. Diagnosticador de decisión

Aplicando las definiciones de la Sección 4.2.1 procedemos a construir el Diagnosticador de decisión para el modelo de la máquina clasificadora de la Figura 14. Este Diagnosticador se compone de ocho estados y puede verse en la Figura 15.

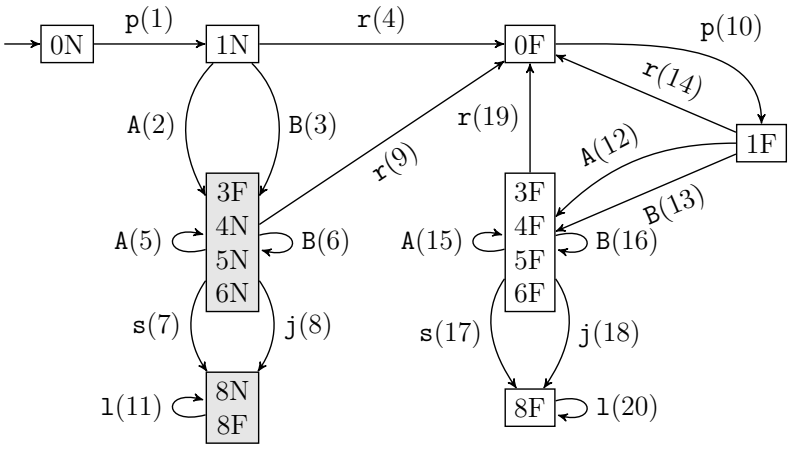


Figura 15: Diagnosticador de decisión del modelo de la Figura 14

Los estados sombreados son los estados ambiguos del Diagnosticador. Las matrices de transiciones del Diagnosticador son las siguientes:

$$(1)\ \begin{bmatrix}1\end{bmatrix} \qquad (2)\ \begin{bmatrix}0{,}05\cdot\pi_3^A & \pi_4^A\cdot 0{,}1425 & \pi_4^A\cdot 0{,}0475 & \pi_4^A\cdot 0{,}76\end{bmatrix}$$

$$(3)\ \begin{bmatrix}0{,}05\cdot\pi_3^B & \pi_4^B\cdot 0{,}285 & \pi_4^B\cdot 0{,}095 & \pi_4^B\cdot 0{,}57\end{bmatrix} \qquad (4)\ \begin{bmatrix}0{,}05\cdot\pi_3^r\end{bmatrix}$$

$$(5)\begin{bmatrix} \pi_3^A & 0 & 0 & 0 \\ 0 & \pi_4^A \cdot 0{,}15 & \pi_4^A \cdot 0{,}05 & \pi_4^A \cdot 0{,}8 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (6)\begin{bmatrix} \pi_3^B & 0 & 0 & 0 \\ 0 & \pi_4^B \cdot 0{,}3 & \pi_4^B \cdot 0{,}1 & \pi_4^B \cdot 0{,}6 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$(7)\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \pi_7^s \\ \pi_6^s & 0 \end{bmatrix} \quad (8)\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \pi_7^j \\ \pi_6^j & 0 \end{bmatrix} \quad (9)\begin{bmatrix} \pi_3^r \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (10)\begin{bmatrix} 1 \end{bmatrix} \quad (11)\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$(12)\begin{bmatrix} 0{,}05 \cdot \pi_3^A & \pi_4^A \cdot 0{,}1425 & \pi_4^A \cdot 0{,}0475 & \pi_4^A \cdot 0{,}76 \end{bmatrix}$$

$$(13)\begin{bmatrix} 0{,}05 \cdot \pi_3^B & \pi_4^B \cdot 0{,}285 & \pi_4^B \cdot 0{,}095 & \pi_4^B \cdot 0{,}57 \end{bmatrix} \quad (14)\begin{bmatrix} 0{,}05 \cdot \pi_3^r \end{bmatrix}$$

$$(15)\begin{bmatrix} \pi_3^A & 0 & 0 & 0 \\ 0 & \pi_4^A \cdot 0{,}15 & \pi_4^A \cdot 0{,}05 & \pi_4^A \cdot 0{,}8 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (16)\begin{bmatrix} \pi_3^B & 0 & 0 & 0 \\ 0 & \pi_4^B \cdot 0{,}3 & \pi_4^B \cdot 0{,}1 & \pi_4^B \cdot 0{,}6 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$(17)\begin{bmatrix} 0 \\ 0 \\ \pi_7^s \\ \pi_6^s \end{bmatrix} \quad (18)\begin{bmatrix} 0 \\ 0 \\ \pi_7^j \\ \pi_6^j \end{bmatrix} \quad (19)\begin{bmatrix} \pi_3^r \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (20)\begin{bmatrix} 1 \end{bmatrix}$$

A partir del Diagnosticador de decisión procedemos a extraer una cadena de Markov para continuar con el análisis de diagnosticabilidad.

6.3.2. Cadena de Markov y su análisis asintótico

El siguiente paso del algoritmo consiste en la extracción de la cadena de Markov a partir de $\mathcal{D}$. Aplicando el procedimiento de la Sección 4.2.4 obtenemos una cadena $\Pi(\mathcal{D})$ que consta de 22 elementos, dados por cada una de las componentes presentes en los estados de $\mathcal{D}$. La matriz de transiciones entre los estados de esta cadena puede encontrarse en la Figura 16.

La cadena de Markov tiene tres clases de recurrencia. Éstas son fáciles de ubicar en la matriz de transiciones, y se corresponden con las componentes

$\Pi(\mathcal{M})$	$0N$	$1N$	$3F$	$4N$	$5N$	$6N$	$0F$	$8N$	$8F$	$1F$	$3F$	$4F$	$5F$	$6F$	$8F$
$0N$	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
$1N$	0	0	$+^{0{,}05\cdot\pi_3^A}_{0{,}05\cdot\pi_3^B}$	$+^{\pi_4^A\cdot 0{,}1425}_{\pi_4^B\cdot 0{,}285}$	$+^{\pi_4^A\cdot 0{,}0475}_{\pi_4^B\cdot 0{,}095}$	$+^{\pi_4^A\cdot 0{,}76}_{\pi_4^B\cdot 0{,}57}$	$0{,}05\cdot\pi_3^r$	0	0	0	0	0	0	0	0
$3F$	0	0	$\pi_3^A+\pi_3^B$	0	0	0	π_3^r	0	0	0	0	0	0	0	0
$4N$	0	0	0	$+^{\pi_4^A\cdot 0{,}15}_{\pi_4^B\cdot 0{,}3}$	$+^{\pi_4^A\cdot 0{,}05}_{\pi_4^B\cdot 0{,}1}$	$+^{\pi_4^A\cdot 0{,}8}_{\pi_4^B\cdot 0{,}6}$	0	0	0	0	0	0	0	0	0
$5N$	0	0	0	0	0	0	0	0	$\pi_7^s+\pi_7^j$	0	0	0	0	0	0
$6N$	0	0	0	0	0	0	0	$\pi_6^s+\pi_6^j$	0	0	0	0	0	0	0
$0F$	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
$8N$	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
$8F$	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
$1F$	0	0	0	0	0	0	$0{,}05\cdot\pi_3^r$	0	0	0	$+^{0{,}05\cdot\pi_3^A}_{0{,}05\cdot\pi_3^B}$	$+^{\pi_4^A\cdot 0{,}1425}_{\pi_4^B\cdot 0{,}285}$	$+^{\pi_4^A\cdot 0{,}0475}_{\pi_4^B\cdot 0{,}095}$	$+^{\pi_4^A\cdot 0{,}76}_{\pi_4^B\cdot 0{,}57}$	0
$3F$	0	0	0	0	0	0	π_3^r	0	0	0	$\pi_3^A+\pi_3^B$	0	0	0	0
$4F$	0	0	0	0	0	0	0	0	0	0	0	$+^{\pi_4^A\cdot 0{,}15}_{\pi_4^B\cdot 0{,}3}$	$+^{\pi_4^A\cdot 0{,}05}_{\pi_4^B\cdot 0{,}1}$	$+^{\pi_4^A\cdot 0{,}8}_{\pi_4^B\cdot 0{,}6}$	0
$5F$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\pi_7^s+\pi_7^j$
$6F$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\pi_6^s+\pi_6^j$
$8F$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figura 16: Cadena de Markov asociada al Diagnosticador de decisión

de los estados $q_4 = \{(8,N),(8,F)\}$ y $q_7 = \{(8,F)\}$ del Diagnosticador de decisión. Denotamos estas clases de recurrencia con $\zeta_1 = \{(q_4,8,N)\}$, $\zeta_2 = \{(q_4,8,F)\}$ y $\zeta_3 = \{(q_7,8,F)\}$.

Separando las filas y columnas correspondientes a clases recurrentes podemos ordenar la matriz de transiciones en su forma canónica compuesta de submatrices Q, R y C, como se explica en la Sección 4.2.5. Haciendo uso de estas matrices y con la ayuda del software matemático SageMath [18] calculamos las matrices $N = (I-Q)^{-1}$ y $B = N \cdot R$. Por cuestiones de espacio, a continuación listamos únicamente las expresiones obtenidas para las primeras filas de las matrices N y B:

$$
\begin{aligned}
N_{(q_0,0,N),(q_0,0,N)} &= 1 \\
N_{(q_0,0,N),(q_1,1,N)} &= 1 \\
N_{(q_0,0,N),(q_2,3,F)} &= -\frac{0{,}05\,\pi_3^A + 0{,}05\,\pi_3^B}{-1+\pi_3^A+\pi_3^B} \\
N_{(q_0,0,N),(q_2,4,N)} &= -\frac{0{,}14\,\pi_4^A + 0{,}28\,\pi_4^B}{-1+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B} \\
N_{(q_0,0,N),(q_2,5,N)} &= \frac{0{,}0002\,{\pi_4^A}^2 - 0{,}048\,\pi_4^A - 0{,}095\,\pi_4^B}{-1.+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B} \\
N_{(q_0,0,N),(q_2,6,N)} &= \frac{0{,}02\,\pi_4^A\,\pi_4^B - 0{,}76\,\pi_4^A - 0{,}57\,\pi_4^B}{-1+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B} \\
N_{(q_0,0,N),(q_3,0,F)} &= -\frac{\left(0{,}05\,\pi_3^A - 0{,}05 + 0{,}05\,\pi_3^B + 0{,}0025\,\pi_3^r\right)\pi_3^r}{\left(-1+0{,}05\,\pi_3^r+\pi_3^A+\pi_3^B\right)^2} \\
N_{(q_0,0,N),(q_5,1,F)} &= -0{,}05\,\frac{\pi_3^r}{-1+0{,}05\,\pi_3^r+\pi_3^A+\pi_3^B} \\
N_{(q_0,0,N),(q_6,3,F)} &= 0{,}05\,\frac{\pi_3^r\left(0{,}05\,\pi_3^A+0{,}05\,\pi_3^B\right)}{\left(-1+\pi_3^A+\pi_3^B\right)\left(-1+0{,}05\,\pi_3^r+\pi_3^A+\pi_3^B\right)} \\
N_{(q_0,0,N),(q_6,4,F)} &= \frac{\left(0{,}001\,{\pi_4^A}^2 + 0{,}0042\,\pi_4^A\,\pi_4^B + 0{,}0042\,{\pi_4^B}^2 - 0{,}007\,\pi_4^A - 0{,}014\,\pi_4^B\right)\pi_3^r}{\left(-1+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B\right)^2\left(-1+0{,}05\,\pi_3^r+\pi_3^A+\pi_3^B\right)} \\
N_{(q_0,0,N),(q_6,5,F)} &= \frac{-0{,}05\,\pi_3^r}{\left(-1+\pi_3^A+\pi_3^B\right)\left(-1+0{,}05\,\pi_3^r\right)\left(-1+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B\right)\left(-1+0{,}05\,\pi_3^r+\pi_3^A+\pi_3^B\right)} \cdot \\
&\quad (\,0{,}048\,\pi_3^A\,\pi_4^A + 0{,}095\,\pi_3^A\,\pi_4^B + 0{,}048\,\pi_3^B\,\pi_4^A + 0{,}095\,\pi_3^B\,\pi_4^B \\
&\quad -0{,}0002\,\pi_3^A\,{\pi_4^A}^2 - 0{,}0002\,\pi_3^B\,{\pi_4^A}^2 + 0{,}0024\,\pi_4^A\,\pi_3^r \\
&\quad +0{,}0048\,\pi_4^B\,\pi_3^r - 0{,}00001\,{\pi_4^A}^2\pi_3^r + 0{,}0002\,{\pi_4^A}^2 \\
&\quad -0{,}0024\,\pi_3^r\,\pi_3^A\,\pi_4^A + 0{,}00001\,\pi_3^r\,\pi_3^A\,{\pi_4^A}^2 - 0{,}0048\,\pi_3^r\,\pi_3^A\,\pi_4^B \\
&\quad -0{,}0024\,\pi_3^r\,\pi_3^B\,\pi_4^A + 0{,}00001\,\pi_3^r\,\pi_3^B\,{\pi_4^A}^2 - 0{,}0048\,\pi_3^r\,\pi_3^B\,\pi_4^B \\
&\quad -0{,}00003\,\pi_4^A\,\pi_3^r\,\pi_4^B - 0{,}048\,\pi_4^A - 0{,}095\,\pi_4^B)
\end{aligned}
$$

$$
\begin{aligned}
N_{(q_0,0,N),(q_6,6,F)} = \; & \frac{\pi_3^r\, 0{,}05}{\left(-1+\pi_3^A+\pi_3^B\right)\left(-1+0{,}05\,\pi_3^r\right)\left(-1+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B\right)\left(-1+0{,}05\,\pi_3^r+\pi_3^A+\pi_3^B\right)}\cdot \\
& (-0{,}001\,\pi_3^r\,\pi_3^A\,\pi_4^A\,\pi_4^B - 0{,}001\,\pi_3^r\,\pi_3^B\,\pi_4^A\,\pi_4^B - 0{,}76\,\pi_3^A\,\pi_4^A \\
& -0{,}57\,\pi_3^A\,\pi_4^B - 0{,}76\,\pi_3^B\,\pi_4^A - 0{,}57\,\pi_3^B\,\pi_4^B - 0{,}038\,\pi_4^A\,\pi_3^r \\
& -0{,}028\,\pi_4^B\,\pi_3^r + 0{,}0002\,{\pi_4^A}^2\pi_3^r + 0{,}0001\,{\pi_4^B}^2\pi_3^r \\
& -0{,}016\,\pi_4^A\,\pi_4^B + 0{,}016\,\pi_3^B\,\pi_4^A\,\pi_4^B + 0{,}038\,\pi_3^r\,\pi_3^A\,\pi_4^A \\
& +0{,}028\,\pi_3^r\,\pi_3^A\,\pi_4^B + 0{,}038\,\pi_3^r\,\pi_3^B\,\pi_4^A + 0{,}028\,\pi_3^r\,\pi_3^B\,\pi_4^B \\
& +0{,}0003\,\pi_4^A\,\pi_3^r\,\pi_4^B + 0{,}016\,\pi_3^A\,\pi_4^A\,\pi_4^B + 0{,}76\,\pi_4^A + 0{,}57\,\pi_4^B)
\end{aligned}
$$

$$
B_{(q_0,0,N),(q_4,8,N)} = \frac{\left(0{,}02\,\pi_4^A\,\pi_4^B - 0{,}76\,\pi_4^A - 0{,}57\,\pi_4^B\right)\left(\pi_6^s+\pi_6^j\right)}{-1+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B}
$$

$$
B_{(q_0,0,N),(q_4,8,F)} = \frac{\left(0{,}0002\,{\pi_4^A}^2 - 0{,}048\,\pi_4^A - 0{,}095\,\pi_4^B\right)\left(\pi_7^s+\pi_7^j\right)}{-1+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B}
$$

$$
\begin{aligned}
B_{(q_0,0,N),(q_7,8,F)} = \; & \frac{-0{,}05\,\pi_3^r\left(\pi_7^s+\pi_7^j\right)}{\left(-1+\pi_3^A+\pi_3^B\right)\left(-1+0{,}05\,\pi_3^r\right)\left(-1+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B\right)\left(-1+0{,}05\,\pi_3^r+\pi_3^A+\pi_3^B\right)}\cdot \\
& (0{,}048\,\pi_3^A\,\pi_4^A + 0{,}095\,\pi_3^A\,\pi_4^B + 0{,}048\,\pi_3^B\,\pi_4^A + 0{,}095\,\pi_3^B\,\pi_4^B \\
& -0{,}0002\,\pi_3^A\,{\pi_4^A}^2 - 0{,}0002\,\pi_3^B\,{\pi_4^A}^2 + 0{,}0024\,\pi_4^A\,\pi_3^r \\
& +0{,}0048\,\pi_4^B\,\pi_3^r - 0{,}00001\,{\pi_4^A}^2\pi_3^r + 0{,}0002\,{\pi_4^A}^2 \\
& -0{,}0024\,\pi_3^r\,\pi_3^A\,\pi_4^A + 0{,}00001\,\pi_3^r\,\pi_3^A\,{\pi_4^A}^2 - 0{,}0048\,\pi_3^r\,\pi_3^A\,\pi_4^B \\
& -0{,}0024\,\pi_3^r\,\pi_3^B\,\pi_4^A + 0{,}00001\,\pi_3^r\,\pi_3^B\,{\pi_4^A}^2 - 0{,}0048\,\pi_3^r\,\pi_3^B\,\pi_4^B \\
& -0{,}00003\,\pi_4^A\,\pi_3^r\,\pi_4^B - 0{,}048\,\pi_4^A - 0{,}095\,\pi_4^B) \\
& + \\
& \frac{0{,}05\,\pi_3^r\left(\pi_6^s+\pi_6^j\right)}{\left(-1+\pi_3^A+\pi_3^B\right)\left(-1+0{,}05\,\pi_3^r\right)\left(-1+0{,}15\,\pi_4^A+0{,}3\,\pi_4^B\right)\left(-1+0{,}05\,\pi_3^r+\pi_3^A+\pi_3^B\right)}\cdot \\
& (-0{,}001\,\pi_3^r\,\pi_3^A\,\pi_4^A\,\pi_4^B - 0{,}001\,\pi_3^r\,\pi_3^B\,\pi_4^A\,\pi_4^B - 0{,}76\,\pi_3^A\,\pi_4^A \\
& -0{,}57\,\pi_3^A\,\pi_4^B - 0{,}76\,\pi_3^B\,\pi_4^A - 0{,}57\,\pi_3^B\,\pi_4^B - 0{,}038\,\pi_4^A\,\pi_3^r \\
& -0{,}028\,\pi_4^B\,\pi_3^r + 0{,}0002\,{\pi_4^A}^2\pi_3^r + 0{,}0001\,{\pi_4^B}^2\pi_3^r \\
& -0{,}016\,\pi_4^A\,\pi_4^B + 0{,}016\,\pi_3^B\,\pi_4^A\,\pi_4^B + 0{,}038\,\pi_3^r\,\pi_3^A\,\pi_4^A \\
& +0{,}028\,\pi_3^r\,\pi_3^A\,\pi_4^B + 0{,}038\,\pi_3^r\,\pi_3^B\,\pi_4^A + 0{,}028\,\pi_3^r\,\pi_3^B\,\pi_4^B \\
& +0{,}0003\,\pi_4^A\,\pi_3^r\,\pi_4^B + 0{,}016\,\pi_3^A\,\pi_4^A\,\pi_4^B + 0{,}76\,\pi_4^A + 0{,}57\,\pi_4^B)
\end{aligned}
$$

A partir de estos valores, podemos obtener expresiones para realizar un análisis sobre los valores de diagnosticabilidad del modelo. Los conjuntos $\zeta_{\neg d}$ y ζ_F están formados de la siguiente manera:

- $\zeta_{\neg d} = \{\zeta_1, \zeta_2\} = \{\{(q_4, 8, N)\}, \{(q_4, 8, F)\}\}$
- $\zeta_F = \{\zeta_2, \zeta_3\} = \{\{(q_4, 8, F)\}, \{(q_7, 8, F)\}\}$

Las ecuaciones para los valores de diagnosticabilidad del modelo son las siguientes:

- $p_{\neg d} = \sum_{\zeta \in \zeta_{\neg d}} \sum_{\lambda \in \zeta} B_{s_0^\Pi, \lambda} = B_{(q_0,0,N),(q_4,8,N)} + B_{(q_0,0,N),(q_4,8,F)}$
- $p_F = \sum_{\zeta \in \zeta_F} \sum_{\lambda \in \zeta} B_{s_0^\Pi, \lambda} = B_{(q_0,0,N),(q_4,8,F)} + B_{(q_0,0,N),(q_7,8,F)}$
- $p_{\neg F} = 1 - p_F = 1 - B_{(q_0,0,N),(q_4,8,F)} + B_{(q_0,0,N),(q_7,8,F)}$
- $p_{F|\neg d} = \dfrac{\sum_{\zeta \in (\zeta_{\neg d} \cap \zeta_F)} \sum_{\lambda \in \zeta} B_{s_0^\Pi, \lambda}}{p_{\neg d}} = \dfrac{B_{(q_0,0,N),(q_4,8,F)}}{B_{(q_0,0,N),(q_4,8,N)} + B_{(q_0,0,N),(q_4,8,F)}}$
- $p_{\neg F|\neg d} = 1 - p_{F|\neg d} = 1 - \dfrac{B_{(q_0,0,N),(q_4,8,F)}}{B_{(q_0,0,N),(q_4,8,N)} + B_{(q_0,0,N),(q_4,8,F)}}$
- $avg_{step} = \sum_{i=1}^{r} N_{s_0^\Pi, \nu_i} =$

$$\begin{array}{l} N_{(q_0,0,N),(q_0,0,N)} + N_{(q_0,0,N),(q_1,1,N)} + N_{(q_0,0,N),(q_2,3,F)} \\ N_{(q_0,0,N),(q_2,4,N)} + N_{(q_0,0,N),(q_2,5,N)} + N_{(q_0,0,N),(q_2,6,N)} \\ N_{(q_0,0,N),(q_3,0,F)} + N_{(q_0,0,N),(q_5,1,F)} + N_{(q_0,0,N),(q_6,3,F)} \\ N_{(q_0,0,N),(q_6,4,F)} + N_{(q_0,0,N),(q_6,5,F)} + N_{(q_0,0,N),(q_6,6,F)} \end{array}$$

A partir de estas ecuaciones podemos formular el problema de optimización.

6.3.3. Problema de optimización

Para responder a las preguntas que nos hicimos en la Sección 6.3, podemos expresarlas como problemas de optimización sobre las expresiones calculadas. Variando la función objetivo podemos obtener diferentes estrategias que resuelvan el no determinismo del MDP del modelo de la máquina clasificadora.

Las variables del problema están dadas por las decisiones que la estrategia debe resolver. Revisando el modelo vemos que no en todos los estados hay que tomar decisiones, sino que esto sucede únicamente en los estados 3, 4, 6 y 7, por lo que las variables del problema son π_3^A, π_3^B, π_3^r, π_4^A, π_4^B, π_6^s, π_6^j, π_7^s y π_7^j. Las restricciones sobre estas variables son las que determinan que sobre cada estado exista una distribuciones de probabilidad sobre los eventos, es decir:

- $0 \leq \pi_3^A, \pi_3^B, \pi_3^r, \pi_4^A, \pi_4^B, \pi_6^s, \pi_6^j, \pi_7^s, \pi_7^j \leq 1$
- $\pi_3^A + \pi_3^B + \pi_3^r = 1$
- $\pi_4^A + \pi_4^B = 1$
- $\pi_6^s + \pi_6^j = 1$
- $\pi_7^s + \pi_7^j = 1$

Variando la función objetivo del problema podemos obtener comportamientos diferentes del sistema. Por ejemplo, para saber si es posible que una máquina clasificadora se comporte de manera tal que tenga la misma probabilidad de fallar que de no hacerlo, basta con setear la función objetivo a $f(X) = \left|p_{F|\neg d} - p_{\neg F|\neg d}\right|$ y minimizar su valor; si la función puede llevarse a 0, entonces dicho comportamiento es posible, y en caso contrario no lo es. Para saber de qué manera debería comportarse una máquina para obtener la mayor y la menor probabilidad de ocurrencia de fallas basta con setear la función objetivo a $f(X) = p_F$ y maximizar/minimizar su valor para obtener las estrategias buscadas. Para conocer los valores de diagnosticabilidad de una estrategia que se comporte uniformemente basta con evaluar las expresiones sobre una estrategia que asigne uniformemente valores a la estrategia y que cumpla con las restricciones.

En la siguiente tabla se resumen los resultados obtenidos para las distintas estrategias buscadas (usamos ε para denotar un valor muy cercano a 0). La solución a estos problemas fue calculada con el módulo para problemas de optimización de SageMath [18].

Estrategias	min $\lvert p_{F\mid\neg d} - p_{\neg F\mid\neg d}\rvert$	min p_F	max p_F	dist. uniforme
$\pi(3)(A)$	0,333333	$0{,}5 - \frac{\varepsilon}{2}$	0,333333	0,333333
$\pi(3)(B)$	0,333333	$0{,}5 - \frac{\varepsilon}{2}$	0,333333	0,333333
$\pi(3)(reset)$	0,333333	ε	0,333333	0,333333
$\pi(4)(A)$	ε	$1 - \varepsilon$	ε	0,5
$\pi(4)(B)$	$1 - \varepsilon$	ε	$1 - \varepsilon$	0,5
$\pi(6)(sidra)$	0,5	0,5	0,5	0,5
$\pi(6)(jugo)$	0,5	0,5	0,5	0,5
$\pi(7)(sidra)$	0,5	0,5	0,5	0,5
$\pi(7)(jugo)$	0,5	0,5	0,5	0,5
$p_{\neg d}$	0,949982	0,949989	0,949982	0,949935
p_F	0,185730	0,106225	0,185730	0,142857
$p_{\neg F}$	0,814270	0,89375	0,814270	0,857143
$p_{F\mid\neg d}$	0,142857	0,05913	0,142857	0,09683
$p_{\neg F\mid\neg d}$	0,857143	0,940827	0,857143	0,902317
$\lvert p_{F\mid\neg d} - p_{\neg F\mid\neg d}\rvert$	0,714286	0,881654	0,714286	0,804635
avg_{step}	3,631598	∞	3,631598	3,489326

Analizando los resultados de esta tabla, podemos concluir lo siguiente:

- La estrategia que minimiza el valor de $\lvert p_{F\mid\neg d} - p_{\neg F\mid\neg d}\rvert$ no logra reducirla a valores cercanos a 0, por lo que no es posible que una máquina tenga las mismas probabilidades de fallar que de no hacerlo.

- La estrategia que minimiza la ocurrencia de fallas en el sistema es aquella que opta por hacer pruebas `A` en el estado 4, pero también la que asigna una chance prácticamente nula a realizar `reset` en el estado 3. Si bien esta es una estrategia posible en teoría, en la práctica es un comportamiento que no debería considerarse. Más aun, con esta estrategia la máquina puede nunca llegar a enterarse el tipo de manzana analizada imposibilitando al controlador central a darle la orden de pasar a clasificar una nueva manzana, ya que una señal `sidra` o `jugo` puede nunca llegar a ser emitida.

Por último aclaramos que si bien en nuestro análisis los valores asignados a $\pi(6)(jugo)$, $\pi(6)(sidra)$, $\pi(7)(jugo)$ y $\pi(7)(sidra)$ no influyen en los cálculos

de los valores de diagnosticabilidad (en este caso el optimizador siempre decidió asignarles el valor 0,5), si pasamos por alto el requerimiento de la Sección 4.2.5 de no truncar transiciones podría darse un comportamiento interesante. Consideremos la estrategia que asigna los siguientes valores:

- $\pi(3)(A) \mapsto 0{,}3$
- $\pi(3)(B) \mapsto 0{,}3$
- $\pi(3)(reset) \mapsto 0{,}4$
- $\pi(4)(A) \mapsto 0{,}5$
- $\pi(4)(B) \mapsto 0{,}5$
- $\pi(6)(sidra) \mapsto 1$
- $\pi(6)(jugo) \mapsto 0$
- $\pi(7)(sidra) \mapsto 0$
- $\pi(7)(jugo) \mapsto 1$

Esta estrategia representa el caso en el que todas las manzanas analizadas deben ser destinadas a la elaboración de sidra, pero siempre que se tome una mala muestra se determina erróneamente que deben ser utilizadas para la elaboración de jugo. Este comportamiento del sistema en particular resulta completamente diagnosticable. Bajo esta estrategia, las fallas del sistema quedan evidenciadas por algún evento observable: la falla `incorrecta_pos` queda evidenciada al observarse un evento `reset` y la falla `mala_muestra` siempre está inmediatamente seguida de un evento `sidra`.

7. Conclusiones

En este trabajo hemos analizado el problema de diagnosticabilidad en modelos que se comportan de manera probabilista. Si bien puede darse una respuesta afirmativa o negativa con respecto a la diagnosticabilidad del sistema, la información aportada por la probabilidad de las transiciones entre los estados permite refinar esta respuesta, obteniendo probabilidades sobre la ocurrencia de trazas no diagnosticables, junto con otros valores de interés.

Para el análisis sobre la diagnosticabilidad en estos modelos, mostramos un método que consiste en la construcción de un Diagnosticador, el cual captura el comportamiento observable de un sistema y permite obtener información sobre ocurrencia de fallas y de trazas no diagnosticables del sistema. Un análisis sobre los valores obtenidos permite al diseñador del sistema decidir si es necesario realizar cambios en el modelo para optimizar su diagnosticabilidad o considerar que el sistema es suficientemente diagnosticable, evitando así gastos innecesarios para incrementar observabilidad o la remodelación del sistema.

El análisis sobre diagnosticabilidad realizado consistió en lo siguiente:

- Comenzamos el análisis sobre LTS como manera de introducción al problema de verificación de diagnosticabilidad de un sistema.
- Analizamos modelos cuyas transiciones están acompañadas por su probabilidad de ocurrencia, analizando diagnosticabilidad sobre pLTS.
- Desarrollamos un nuevo método aplicable a sistemas que mezclan no determinismo junto con distribuciones de probabilidad, los cuales fueron modelados como procesos de decisión de Markov. Incluso, un análisis similar puede realizarse sobre autómatas probabilistas.
- Además mostramos como el Diagnosticador puede ser utilizado para realizar diagnosis.

Inspirados por el método "Twin-Plant", el cual redujo considerablemente la complejidad de la verificación de diagnosticabilidad con respecto a algoritmos previos, como lineas de trabajos futuros nos interesa mejorar el método desarrollado de manera tal que el análisis de diagnosticabilidad de un sistema no necesite de un Diagnosticador.

También planeamos considerar el análisis de diagnosticabilidad sobre otros tipos de formalismos probabilísticos como redes de Petri probabilistas y

autómatas probabilísticos, en sistemas de tiempo real y en sistemas distribuidos.

Otras áreas que nos interesan son la verificación y análisis de distinguibilidad y predecibilidad sobre sistemas probabilísticos.

Apéndices

A. Optimización con restricciones lineales

A.1. Introducción

En esta sección hacemos una breve introducción al problema de optimización de funciones con restricciones lineales, lo cual es de vital importancia para completar el análisis de diagnosticabilidad sobre MDP. Para esto, presentamos un resumen del algoritmo publicado en [1].

Este algoritmo sirve para resolver problemas de optimización donde las restricciones están dadas por funciones lineales que determinan una región acotada de posibles soluciones, y cuya función objetivo es una función continua diferenciable en casi todas partes, es decir, con una cantidad finita de puntos donde no es diferenciable. El esquema de estos problemas es el siguiente:

$$\begin{aligned} \underset{X}{\text{maximizar}} \quad & f(X) \\ \text{sujeto a} \quad & g_i(X) \leq b_i, \quad i = 1, \ldots, m \\ & h_i(X) = c_i, \quad i = 1, \ldots, p \end{aligned}$$

donde $f : \mathbb{R}^n \to \mathbb{R}$ es la *función objetivo*, la cual es continua y diferenciable en casi todos lados, $g_i(X)$ y $h_i(X)$ son restricciones lineales sobre las posibles soluciones, las cuales determinan una región acotada de soluciones factibles para el problema conocida como *región factible*.

El algoritmo se basa en la evaluación de la función objetivo en puntos críticos, ya sea estrictamente dentro de la región factible como así también en sus caras, aristas y vértices. Se lo puede resumir en los siguientes pasos:

1. Encontrar los vértices que determinan la región factible.
2. Encontrar los puntos críticos de la función objetivo usando su gradiente y verificar si están dentro de la región factible.
3. Computar todos los puntos críticos presentes en las caras y aristas de la región factible restringiendo la función objetivo a los dominios dados por dichas caras y aristas.

4. Evaluar la función objetivo en todos los vértices y puntos críticos encontrados y elegir la solución óptima.

A.2. Identificación de vértices, aristas y caras

El interior de la región factible está definido por el conjunto de vértices que lo determinan. Las caras y aristas de la región factible están dadas por ciertos subconjuntos de estos vértices.

A.2.1. Vértices

Las coordenadas de los vértices de la región factible se obtienen resolviendo los sistemas de ecuaciones que resultan de tomar algunas restricciones y cambiar las desigualdades por igualdades. El número de vértices de la región factible es a lo sumo $\binom{r}{q}$, donde q es la cantidad de variables del problema y r es la cantidad de restricciones. Por lo tanto, si tomamos de a q ecuaciones y resolvemos el sistema obtenemos una posible solución, y si dicha solución cumple el resto de las restricciones, entonces es un vértice de la región factible.

Ejemplo: Supongamos que queremos encontrar los vértices de la región factible dada por las siguientes restricciones:

$$\begin{aligned} 3x_1 + 2x_2 &\leq 6 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Esta región tiene a lo sumo $\binom{3}{2} = 3$ vértices, los cuales se obtienen resolviendo los siguientes sistemas de ecuaciones:

$$\begin{cases} 3x_1 + 2x_2 = 6 \\ x_2 = 0 \end{cases} \qquad \begin{cases} 3x_1 + 2x_2 = 6 \\ x_1 = 0 \end{cases} \qquad \begin{cases} x_1 = 0 \\ x_2 = 0 \end{cases}$$

Las soluciones a estos sistemas son $(2, 0)$, $(0, 3)$ y $(0, 0)$ respectivamente. Como estas soluciones cumplen con todas las restricciones del problema, son también los vértices de la región factible.

A.2.2. Aristas y caras

El conjunto de aristas, y caras (de distintas dimensiones) están dadas por subconjuntos de vértices que cumplen simultáneamente varias restricciones. A continuación presentamos un método para determinar dichos subconjuntos de vértices usando una tabla de restricciones-vértices.

Sea n el número de variables del problema, construimos una tabla con una columna por cada vértice de la región factible y una fila por cada una de las restricciones consideradas como igualdades. Luego agrupamos conjuntos de vértices que cumplen simultáneamente algunas restricciones de la siguiente manera:

- agrupamos los vértices que cumplen cada restricción. Cada conjunto de vértices que satisface cada restricción define una cara en el caso tridimensional o una arista en el caso bidimensional,
- agrupamos los vértices que cumplen de a dos restricciones simultáneamente. Cada conjunto de vértices que cumplen las mismas restricciones definen aristas en el caso tridimensional,
- agrupamos los vértices que cumplen de a tres, cuatro, ..., $n-1$ restricciones para obtener el resto de las caras en otras dimensiones.

Ejemplo: Consideremos la siguiente región factible:

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 10 \\ 3x_1 + x_3 &\leq 24 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

La región factible se puede ver en la Figura 17.

Los **vértices** de la región factible pueden ser calculados como vimos en la sección anterior, y son los siguientes:

- $v_1 = (8, 0, 0)$
- $v_2 = (8, 2, 0)$
- $v_3 = (0, 10, 0)$
- $v_4 = (0, 0, 10)$
- $v_5 = (7, 0, 3)$
- $v_6 = (0, 0, 0)$

Construimos la tabla con las restricciones que cumple cada vértice, tomando las restricciones como igualdades:

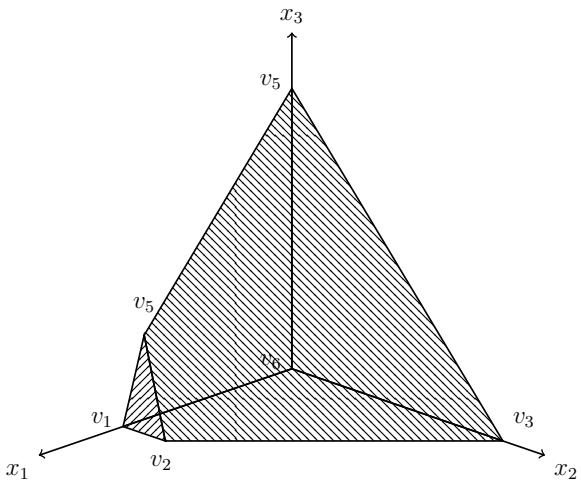


Figura 17: Región factible del problema

Restricción	v_1	v_2	v_3	v_4	v_5	v_6
$x_1 + x_2 + x_3 = 10$	no	si	si	si	si	no
$3x_1 + x_3 = 24$	si	si	no	no	si	no
$x_1 = 0$	no	no	si	si	no	si
$x_2 = 0$	si	no	no	si	si	si
$x_3 = 0$	si	si	si	no	no	si

Las **caras** de la región factible están dadas por los vértices que cumplen cada restricción:

Restricción en común	Vértices
$x_1 + x_2 + x_3 = 10$	v_2, v_3, v_4, v_5
$3x_1 + x_3 = 24$	v_1, v_2, v_5
$x_1 = 0$	v_3, v_4, v_6
$x_2 = 0$	v_1, v_4, v_5, v_6
$x_3 = 0$	v_1, v_2, v_3, v_6

Por lo tanto la región factible tiene 5 caras, cada una delimitada por cada grupo de vértices de la tabla.

Las **aristas** de la región factible están dadas por los vértices que cumplen simultáneamente dos restricciones:

Restricción 1	Restricción 2	Vértices
$x_1 + x_2 + x_3 = 10$	$3x_1 + x_3 = 24$	v_2, v_5
$x_1 + x_2 + x_3 = 10$	$x_1 = 0$	v_3, v_4
$x_1 + x_2 + x_3 = 10$	$x_2 = 0$	v_4, v_5
$x_1 + x_2 + x_3 = 10$	$x_3 = 0$	v_2, v_4
$3x_1 + x_3 = 24$	$x_1 = 0$	Ninguno
$3x_1 + x_3 = 24$	$x_2 = 0$	v_1, v_5
$3x_1 + x_3 = 24$	$x_3 = 0$	v_1, v_2
$x_1 = 0$	$x_2 = 0$	v_4, v_6
$x_1 = 0$	$x_3 = 0$	v_3, v_6
$x_2 = 0$	$x_3 = 0$	v_1, v_6

Por lo tanto la región factible tiene 9 aristas que unen a cada uno de los pares de vértices de la tabla.

En un poliedro, si F es el número de caras, E el número de aristas y V el número de vértices entonces debe darse que $F + V = E + 2$. En nuestro ejemplo tenemos que $5 + 6 = 9 + 2$, lo cual cumple la fórmula.

A.3. Representación paramétrica

La región factible está determinada por sus vértices. Cada punto dentro de la región factible puede ser expresado como una combinación convexa de los vértices. Una *combinación convexa* de los vértices $v_1, v_2, \ldots, v_m$ está dada por:

$$\left\{ \sum_{i=1}^{m} \lambda_i v_i : 0 \leq \lambda_i \leq 1 \wedge \sum_{i=1}^{m} \lambda_i = 1 \right\}$$

. Por ejemplo, la arista que une los vértices v_i y v_j está dada por el conjunto $\{\lambda v_i + (1 - \lambda)v_j, 0 \leq \lambda \leq 1\}$. Llamamos *representación paramétrica* a la representación de una región como una combinación de sus vértices.

Ejemplo: Queremos representar la región factible definida por las siguientes restricciones en forma paramétrica:

$$\begin{aligned} -3x_1 + x_2 &\leq 6 \\ x_1 + 2x_2 &\leq 4 \\ x_2 &\geq 3 \end{aligned}$$

Los vértices de esta región son los siguientes:

- $v_1 = (-3, -3)$
- $v_1 = (10, -3)$
- $v_1 = (-\frac{8}{7}, \frac{18}{7})$

La representación paramétrica de la región factible con parámetros λ_1, λ_2 y λ_3 para el primer, segundo y tercer vértice es

$$\{(x_1, x_2) \,|\, x_1 = -3\lambda_1 + 10\lambda_2 - \frac{8}{7}\lambda_3 \wedge x_2 = -3\lambda_1 - 3\lambda_2 + \frac{18}{7}\lambda_3\}$$

para todos los $\lambda_1, \lambda_2, \lambda_3 \geq 0$ tales que $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Cualquier punto de la región factible puede obtenerse asignando valores para λ_1, λ_2 y λ_3.

A.4. Cálculo de la solución óptima

Para computar el punto óptimo de la región factible procedemos de la siguiente manera:

Paso 1: Encontrar los vértices que determinan la región factible resolviendo $\binom{r}{q}$ sistemas de ecuaciones.

Paso 2: Computar puntos críticos interiores. Para ello computar el gradiente de la función objetivo y analizar los puntos donde se anula o donde está indefinido. Descartar aquellos puntos que no cumplan con las restricciones.

Paso 3: Computar puntos críticos en caras y aristas de la región factible. Para ello expresamos las caras y aristas en forma paramétrica y buscamos puntos críticos calculando el gradiente y analizando donde se anula o queda indefinido usando la regla de la cadena.

Paso 4: Evaluar la función objetivo en los vértices y puntos críticos encontrados. Elegir la solución óptima.

Ejemplo: Obtener una solución óptima para el siguiente problema:

$$\begin{aligned} \underset{X}{\text{maximizar}} \quad & f(X) = 5x_1 - x_1^2 + 8x_2 - 2x_2^2 \\ \text{sujeto a} \quad & 3x_1 + 2x_2 \leq 6 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

Paso 1: Notar que las restricciones son similares a las restricciones del ejemplo del cálculo de vértices, por lo que los vértices de la región factible son:

- $v_1 = (2, 0)$
- $v_2 = (0, 3)$
- $v_3 = (0, 0)$

Paso 2: El gradiente de la función objetivo es $\nabla f(X) = (5 - 2x_1, 8 - 4x_2)$, la cual se anula en el punto $(\frac{5}{2}, 2)$, el cual no cumple con la restricción $3x_1 + 2x_2 \leq 6$. Por lo tanto no hay puntos críticos estrictamente en el interior de la región factible.

Paso 3: La región factible contiene tres aristas:

Arista 1: Esta arista está dada por el segmento que une los vértices $v_1 = (2, 0)$ y $v_2 = (0, 3)$. Usando la combinación convexa de los vértices damos la representación paramétrica del segmento:

$$\{(x_1, x_2) \,|\, x_1 = 2\lambda_1 \wedge x_2 = 3 \underbrace{\lambda_2}_{(1-\lambda_1)}\}$$

La derivada de $f(X)$ con respecto a λ_1 es:

$$f'(\lambda_1) = \sum_{i=1}^{2} \frac{\partial f(X)}{\partial x_i} \cdot \frac{\partial x_i}{\partial \lambda_1} = (5-2x_1)(2)+(8-4x_2)(-3) = 22-44\lambda_1$$

La derivada se anula en $\lambda_1 = \frac{1}{2}$, lo cual se corresponde con el punto $c_1 = (1, \frac{3}{2})$, que cumple con las restricciones del problema.

Arista 2: Esta arista está dada por el segmento que une los vértices $v_1 = (2, 0)$ y $v_3 = (0, 0)$. Su representación paramétrica es:

$$\{(x_1, x_2) \,|\, x_1 = 2\lambda_1 \wedge x_2 = 0\}$$

La derivada de $f(X)$ con respecto a λ_1 es:

$$f'(\lambda_1) = \sum_{i=1}^{2} \frac{\partial f(X)}{\partial x_i} \cdot \frac{\partial x_i}{\partial \lambda_1} = (5-2x_1)(2)+(8-4x_2)(0) = 10-8\lambda_1$$

la cual se anula cuando $\lambda_1 = \frac{5}{4}$, lo cual se corresponde con el punto $c_2 = (\frac{5}{2}, 0)$, el cual no cumple con la restricción $3x_1 + 2x_2 \leq 6$.

Arista 3: Esta arista está dada por el segmento que une los vértices $v_2 = (0, 3)$ y $v_3 = (0, 0)$. Su representación paramétrica es:

$$\{(x_1, x_2) \,|\, x_1 = 0 \wedge x_2 = 3\lambda_1\}$$

La derivada de $f(X)$ con respecto a λ_1 es:

$$f'(\lambda_1) = \sum_{i=1}^{2} \frac{\partial f(X)}{\partial x_i} \cdot \frac{\partial x_i}{\partial \lambda_1} = (5-2x_1)(0)+(8-4x_2)(3) = 24-36\lambda_1$$

La derivada se anula en $\lambda_1 = \frac{2}{3}$, lo cual se corresponde con el punto $c_3 = (0, 2)$, que cumple con las restricciones del problema.

Paso 4: El resultado de evaluar la función en los vértices y puntos críticos de la región crítica es:

- $f(v_1) = 4$
- $f(v_2) = 6$
- $f(v_3) = 0$
- $f(c_1) = 11{,}5$
- $f(c_3) = 8$

Si comparamos estos resultados, llegamos a la conclusión de que la solución óptima se da en el punto $c_1 = (1, \frac{3}{2})$, maximizando la función objetivo al valor $f(c_1) = 11{,}5$.

Referencias

[1] Hossein Arsham, Miro Gradisar, and Mojca Indihar Stemberger. Linearly constrained global optimization: a general solution algorithm with applications. *Applied Mathematics and Computation*, 134(2–3):345 – 361, 2003.

[2] Nathalie Bertrand, Eric Fabre, Stefan Haar, Serge Haddad, and Loïc Hélouët. Active diagnosis for probabilistic systems. In *FOSSACS'14*, pages 29–42, 2014.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[4] Laura Brandán Briones, Agnes Madalinski, and Hernán Ponce de León. Distributed diagnosability analysis with Petri nets. In Rui Abreu, Ingo Pill, and Franz Wotawa, editors, *Proceedings of the 25th International Workshop on Principles of Diagnosis (DX'14)*, Graz, Austria, September 2014.

[5] Laura Brandán Briones, Alexander Lazovik, and Philippe Dague. Optimizing the system observability level for diagnosability. In *ISoLA'08*, pages 815–830, 2008.

[6] Alessandro Cimatti, Charles Pecheur, and Roberto Cavada. Formal verification of diagnosability via symbolic model checking. In *In Proceedings of the 18th International Joint Conference on Artificial Intelligence IJCAI'03*, pages 363–369, 2003.

[7] Hernán Ponce de León, Gonzalo Bonigo, and Laura Brandán Briones. Distributed analysis of diagnosability in concurrent systems. In Meir Kalech, Alexander Feldman, and Gregory Provan, editors, *Proceedings of the 24th International Workshop on Principles of Diagnosis (DX'13)*, Jerusalem, Israel, 2013.

[8] Vojtech Forejt, Marta Kwiatkowska, Gethin Norman, and David Parker. Automated Verification Techniques for Probabilistic Systems. In M. Bernardo and V. Issarny, editors, *Formal Methods for Eternal Networked Software Systems (SFM'11)*, volume LNCS 6659, pages 53–113. Springer, 2011.

[9] DoritS. Hochbaum. Complexity and algorithms for nonlinear optimization problems. *Annals of Operations Research*, 153(1):257–296, 2007.

[10] Paul G. Hoel, Sidney C. Port, and Charles J. Stone. *Introduction to Stochastic Processes*. Houghton Mifflin Company, Boston, MA, 1972.

[11] Xiaowei Huang. Diagnosability in concurrent probabilistic systems. In *AAMAS'13*, pages 853–860, 2013.

[12] Shengbing Jiang, Zhongdong Huang, Vigyan Chandra, and Ratnesh Kumar. A polynomial algorithm for testing diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46:1318–1321, 2000.

[13] A. Madalinski and V. Khomenko. Diagnosability verification with parallel ltl-x model checking based on petri net unfoldings. In *Control and Fault-Tolerant Systems (SysTol), 2010 Conference on*, pages 398–403, Oct 2010.

[14] Dmitry Myadzelets and Andrea Paoli. Virtual modules in discrete-event systems: Achieving modular diagnosability. *CoRR*, abs/1311.2850, 2013.

[15] Farid Nouioua and Philippe Dague. A probabilistic analysis of diagnosability in discrete event systems. In *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pages 224–228, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[16] M. Sampath, Raja Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *Automatic Control, IEEE Transactions on*, 40(9):1555–1575, Sep 1995.

[17] Anika Schumann and Yannick Pencolé. Scalable diagnosability checking of event-driven system. In *In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI07*, pages 575–580, 2007.

[18] W. A. Stein et al. *Sage Mathematics Software (Version 6.7)*. The Sage Development Team, 2015. `http://www.sagemath.org`.

[19] David Thorsley and Demosthenis Teneketzis. Diagnosability of stochastic discrete-event systems. *IEEE Trans. Autom. Control*, pages 476–492, 2005.

[20] Tae-Sic Yoo and Stephane Lafortune. Polynomial-time verification of diagnosability of partially-observed discrete-event systems, 2002.