Minería de Procesos: Descubrimiento y Conformidad

Minería de Procesos y Planificación Automática
Máster en Ciencia de Datos e Ingeniería de Computadores

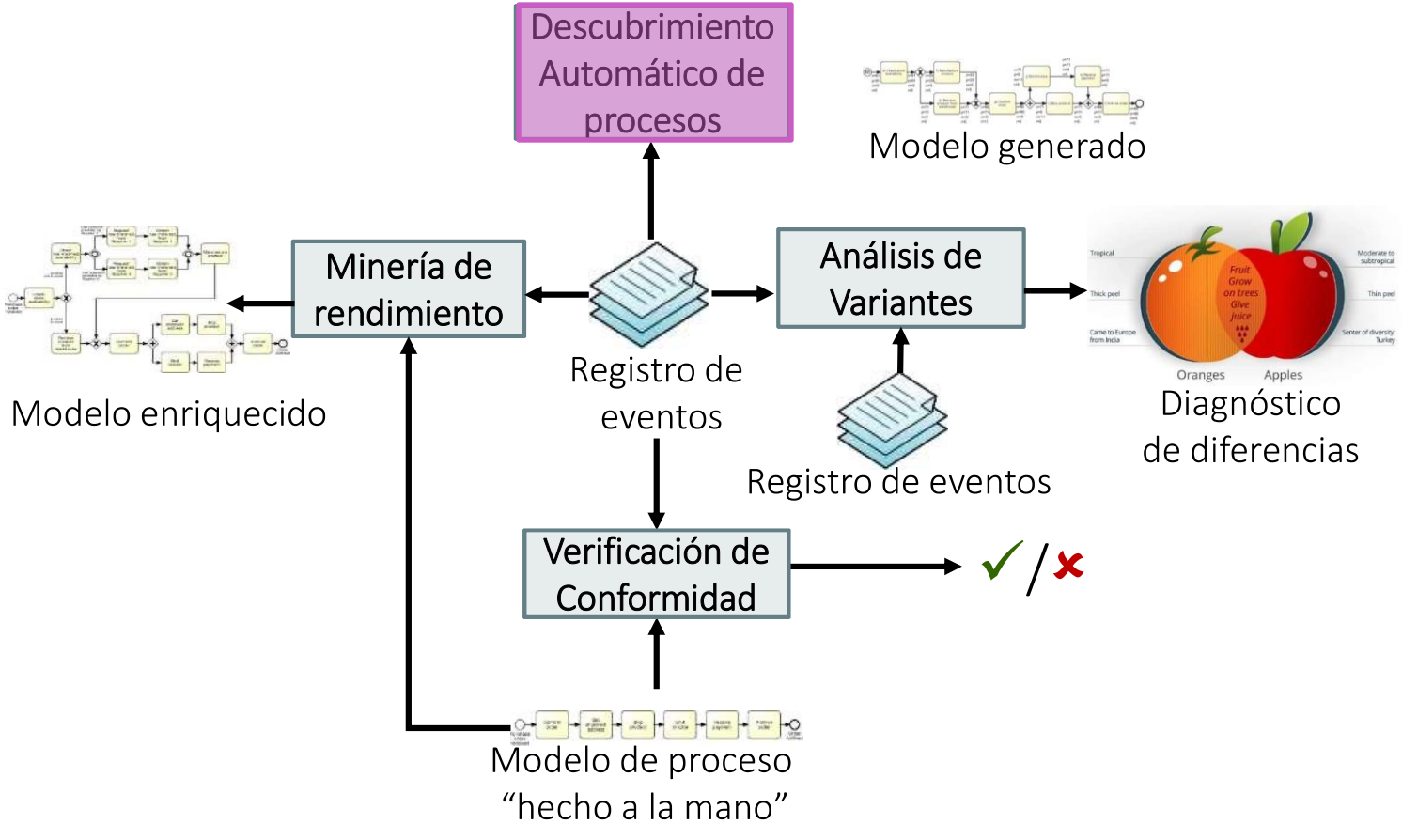
Adela del Río Ortega, Abril 2025

En la clase anterior

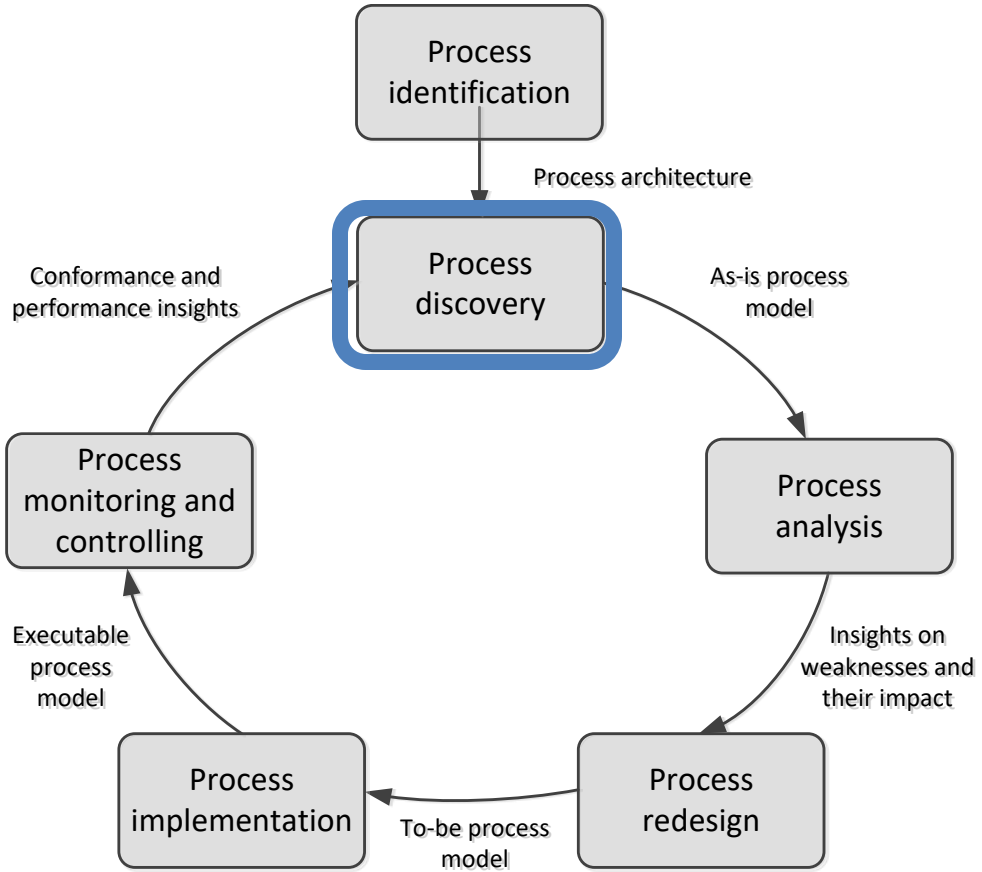
Procesos de negocio
Ciclo de vida de los procesos de negocio
Minería de procesos
Descubrimiento automático de procesos
Herramientas
Ejercicios
Operaciones sobre registros de eventos
Ruidos en registros de eventos
Limitaciones de los mapas de procesos

Descubrimiento automático de procesos

Casos de uso


Descubrimiento Automático de procesos
Modelo generado
Minería de rendimiento
Modelo enriquecido
Registro de eventos
Análisis de Variantes
Registro de eventos
Diagnóstico de diferencias
Verificación de Conformidad
✓ / ✗
Modelo de proceso "hecho a la mano"

El ciclo de vida de los procesos de negocio



Fase 2: Descubrimiento de procesos (proceso “as is”)

El descubrimiento de procesos busca proporcionar mayor comprensión sobre el proceso. Es el acto de recopilar información sobre un proceso existente y organizarlo en términos de un modelo de proceso.

Criterios para comenzar el descubrimiento

- Importancia
- Disfunción
- Viabilidad

El descubrimiento de procesos es una actividad colaborativa

	Analista de procesos	Experto de dominio
Habilidades de modelado	Alta	Limitada
Conocimiento del proceso	Limitado	Alto

Dificultades en el descubrimiento de procesos

Conocimiento fragmentado del proceso

Expertos de dominio piensan en casos concretos

Expertos de dominio no están familiarizados con notaciones de procesos

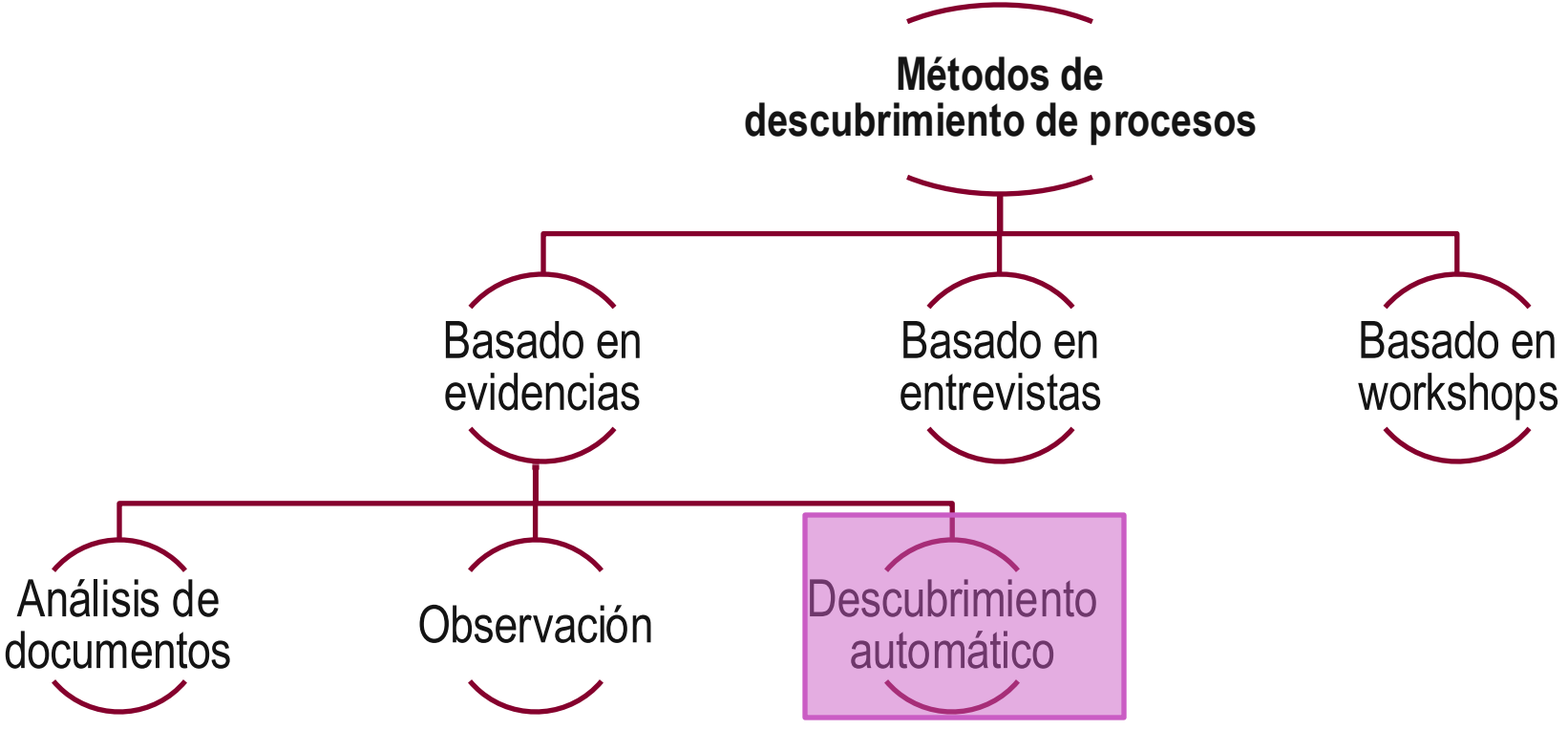
Metodos de descubrimiento de procesos

EVIDENCIAS

ENTREVISTAS

WORKSHOPS

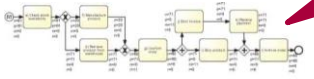
Métodos de descubrimiento de procesos



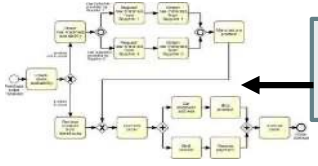
Casos de uso

Capturar el comportamiento del registro de manera representativa

Descubrimiento Automático de procesos



Modelo generado



Modelo enriquecido

Minería de rendimiento

Registro de eventos

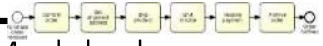
Análisis de Variantes

Diagnóstico de diferencias

Registro de eventos

Verificación de Conformidad

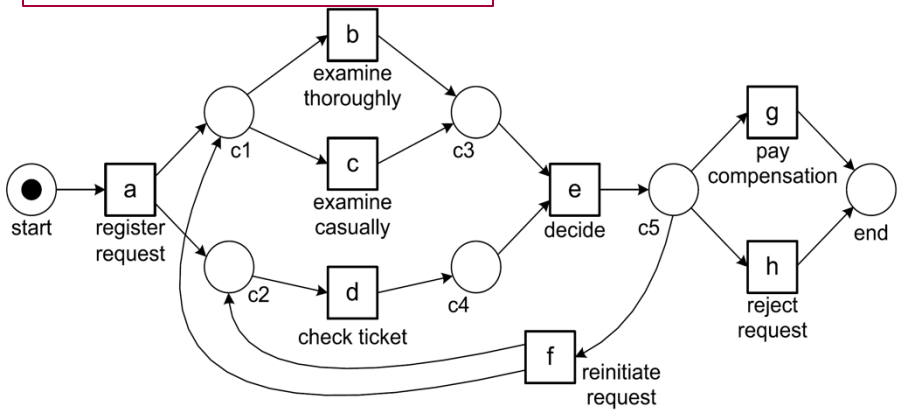
✓ / ✗



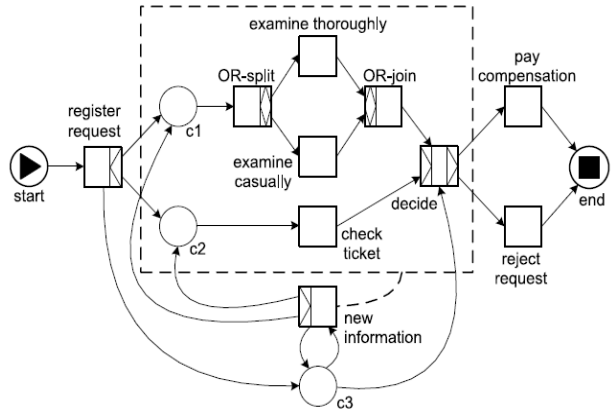
Modelo de proceso “hecho a la mano”

Representaciones de procesos

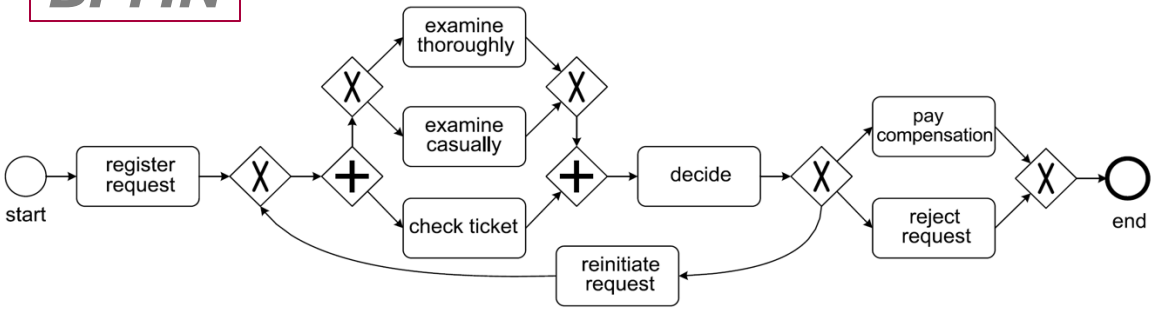
Redes de Petri



YAWL



BPMN



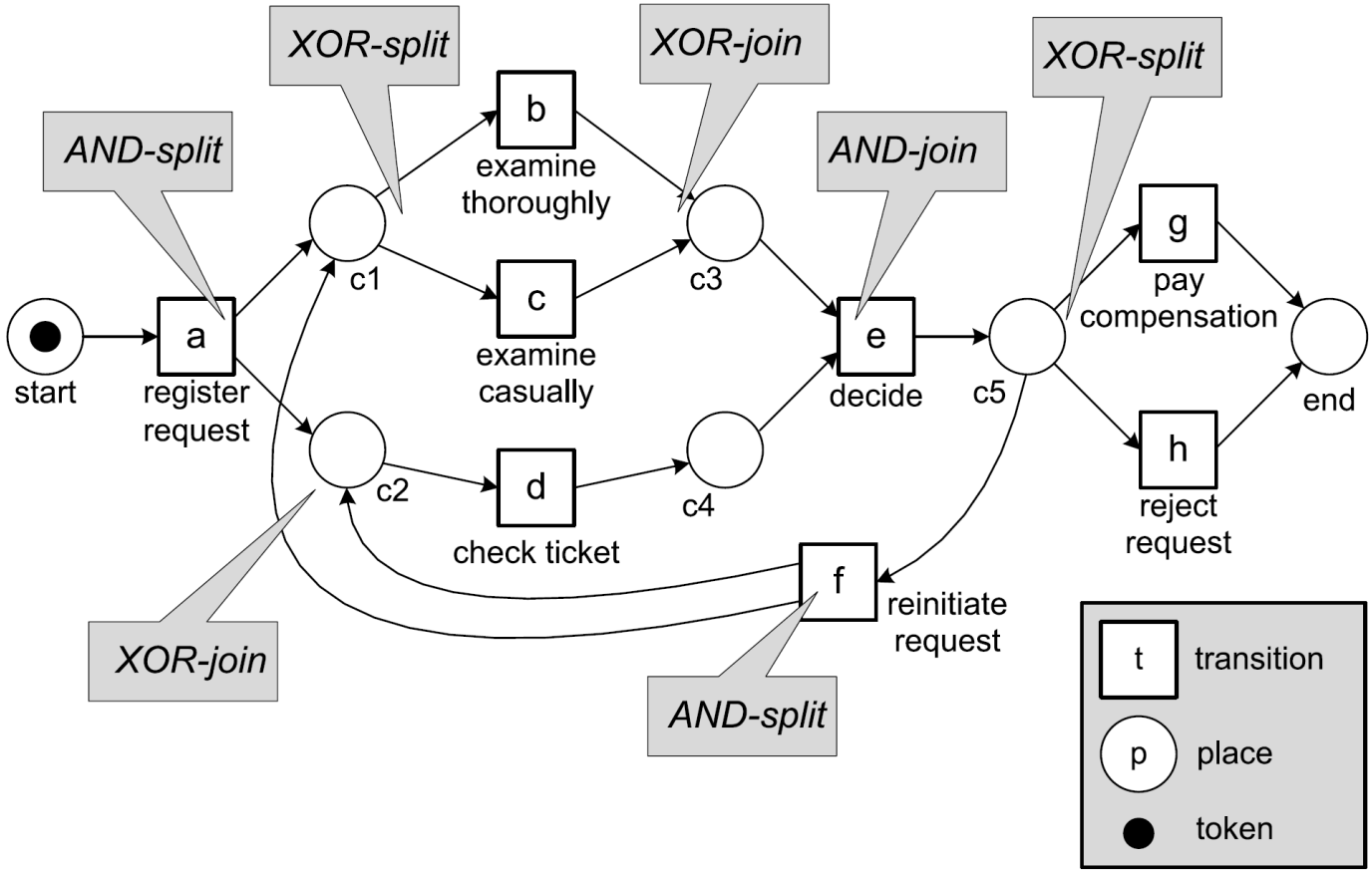
EPCs

Causal Nets

Process Trees

Process Mining: Data Science in Action. Springer Berlin, Heidelberg, 2016.

Redes de Petri

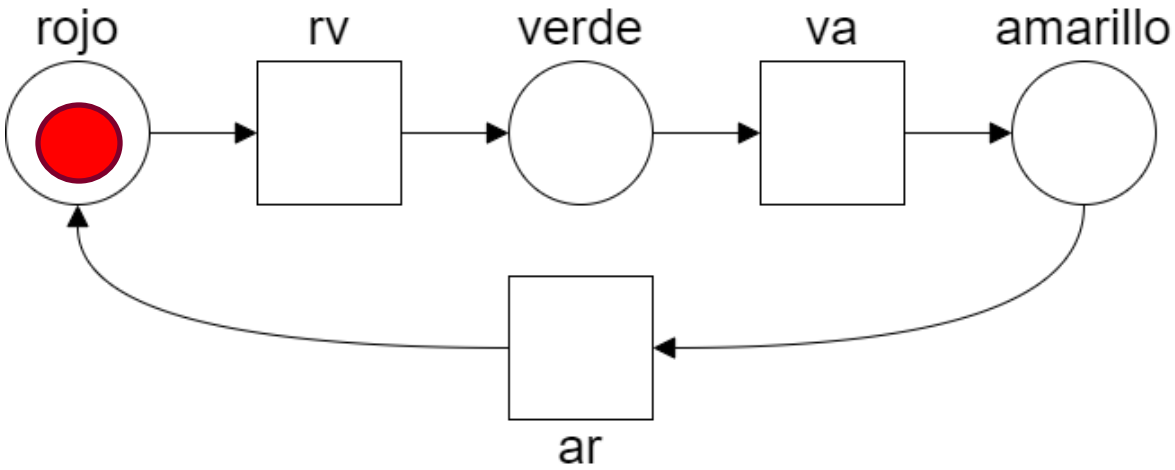


Process Mining: Data Science in Action. Springer Berlin, Heidelberg, 2016.

Redes de Petri

Un proceso muy simple:

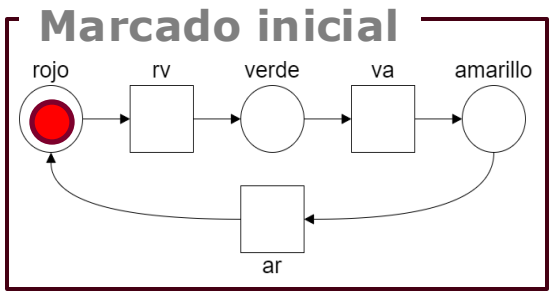
- Un semáforo puede estar en 3 estados posibles.
- Entre cada par de estados hay una transición.

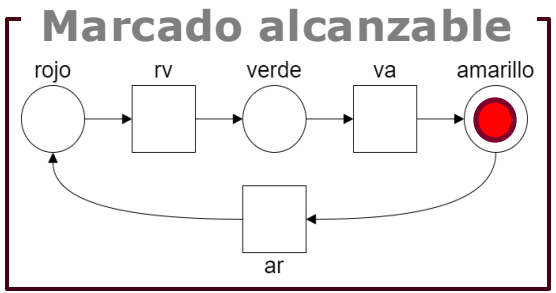


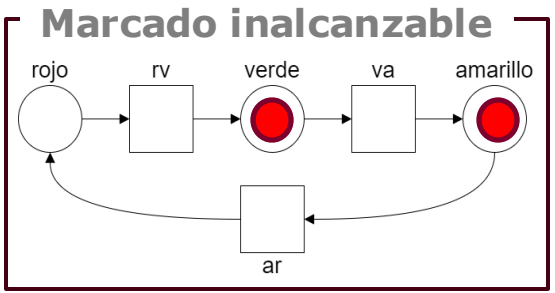
Process Mining: Data Science in Action. Springer Berlin, Heidelberg, 2016.

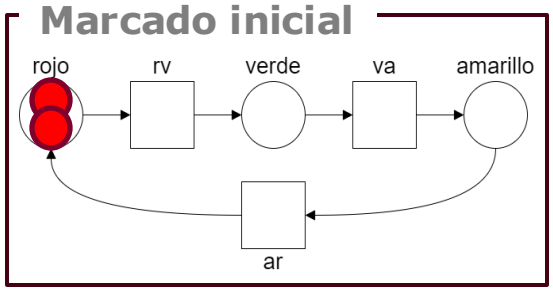
Redes de Petri: Marcado

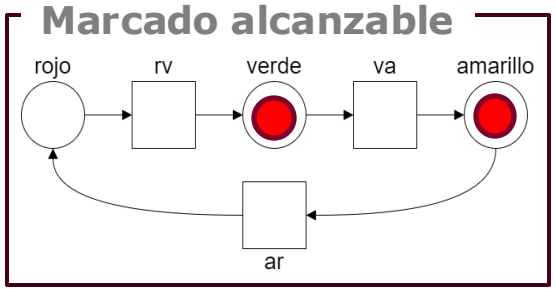
Marcado: Es una instantánea de la red de Petri.

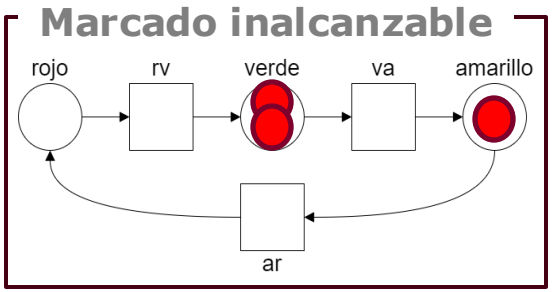








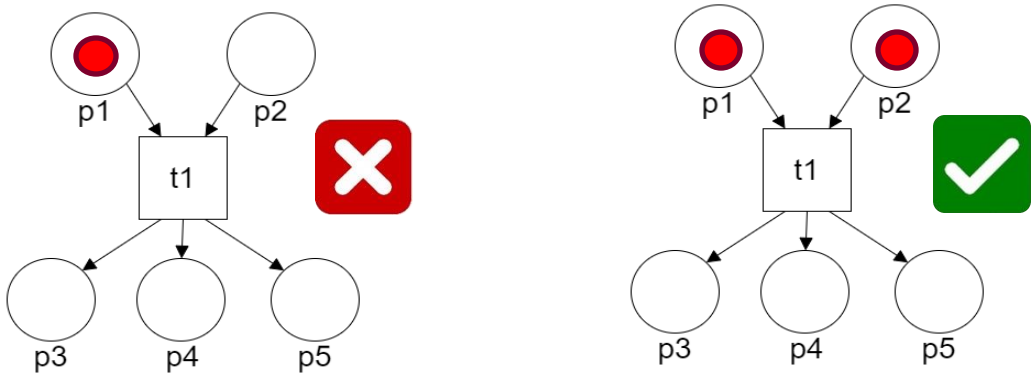




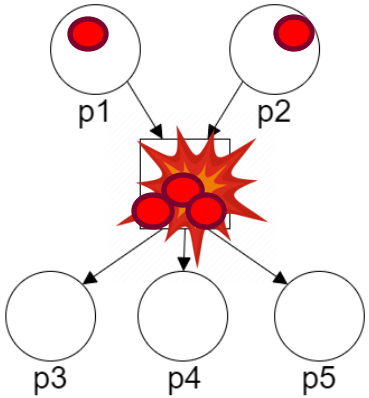
Process Mining: Data Science in Action. Springer Berlin, Heidelberg, 2016.

Redes de Petri: Marcado

Activación: Una transición está activada si cada lugar de entrada contiene un token.

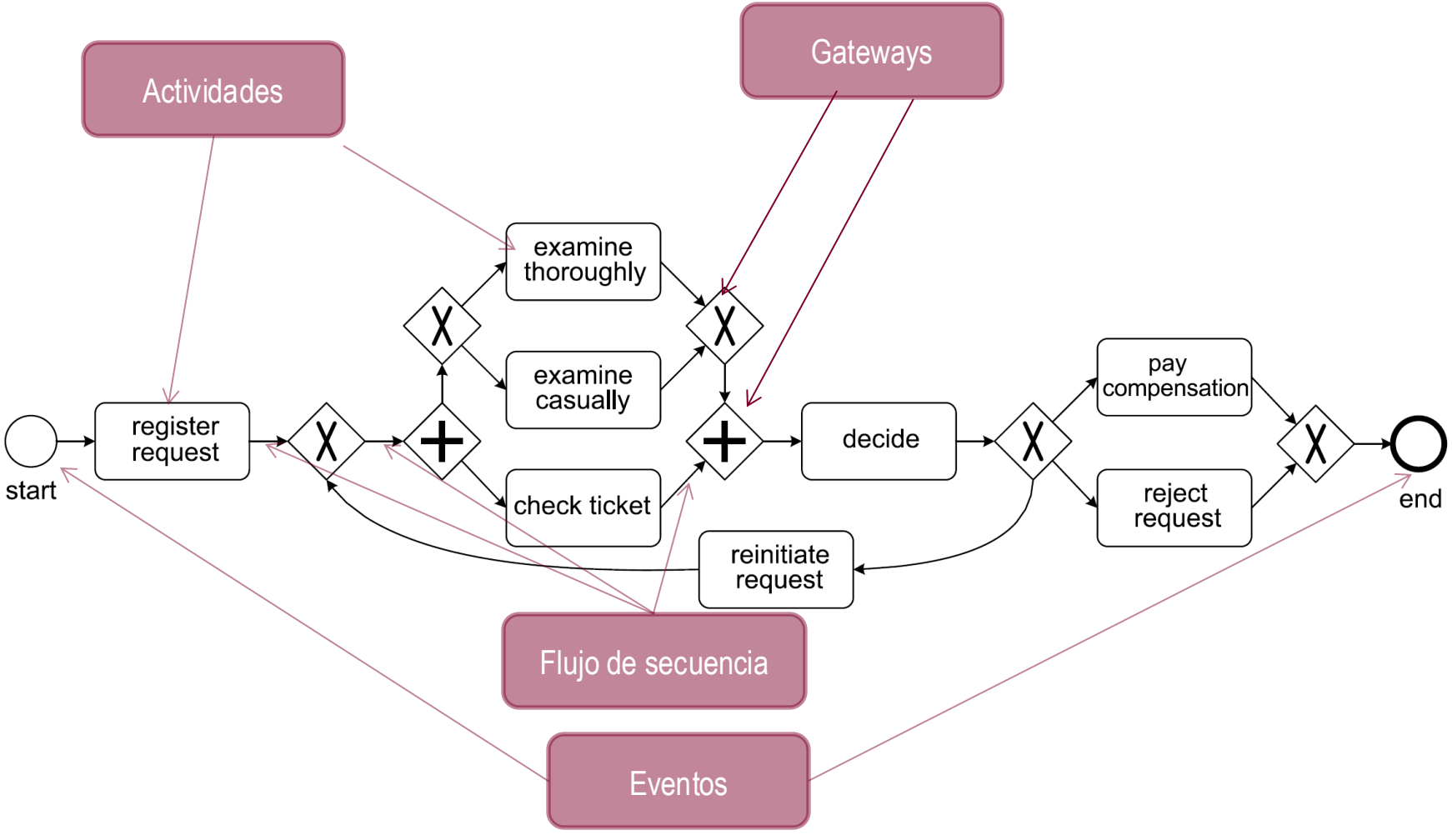


Disparo: Una transición activada puede dispararse consumiendo un token desde cada lugar de entrada y produciendo un token para cada lugar de salida.

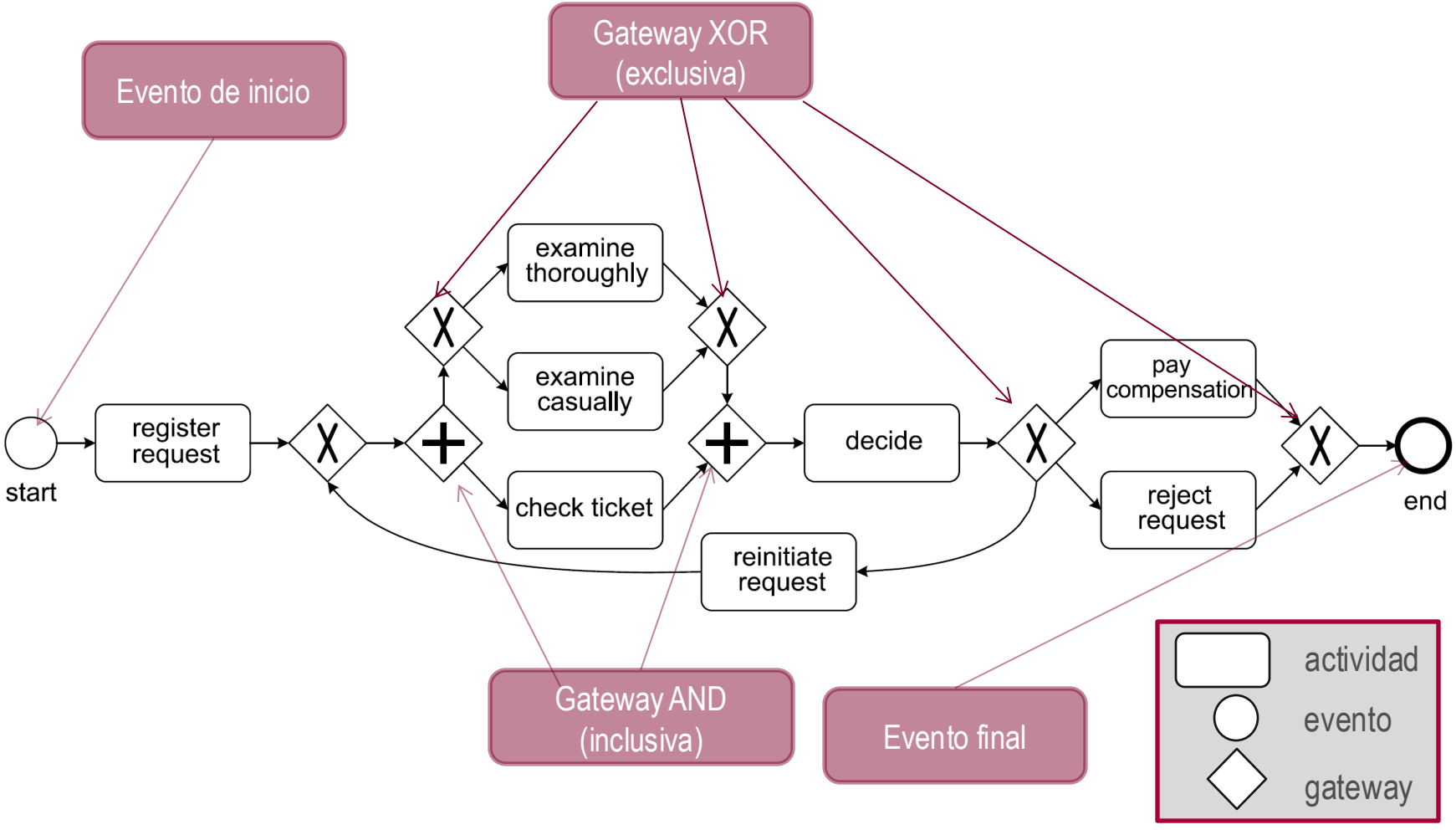


Process Mining: Data Science in Action. Springer Berlin, Heidelberg, 2016.

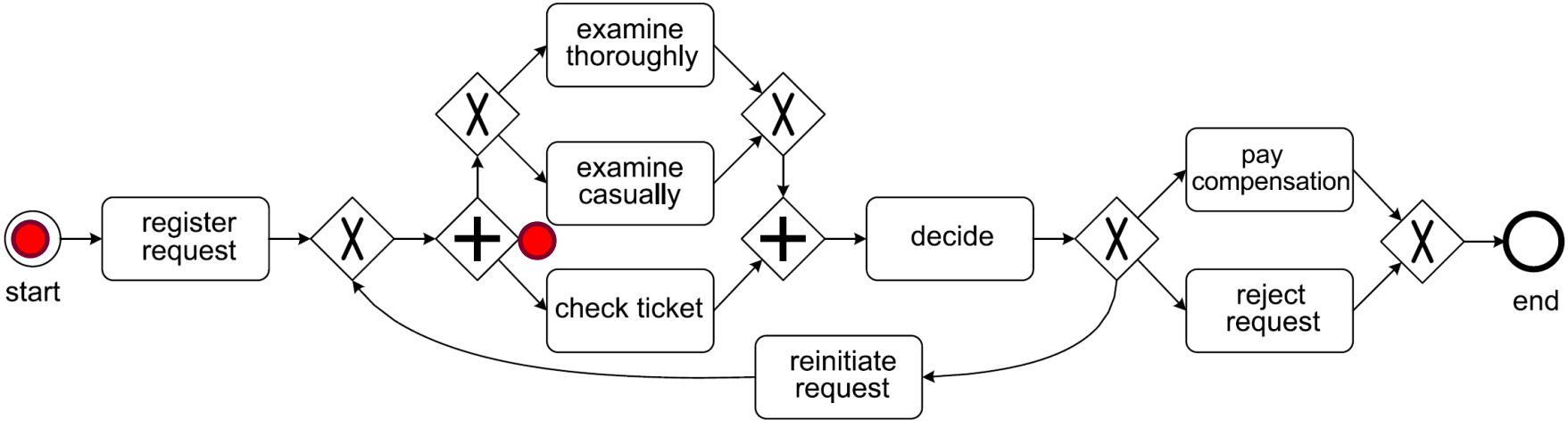
BPMN (Business Process Model and Notation)

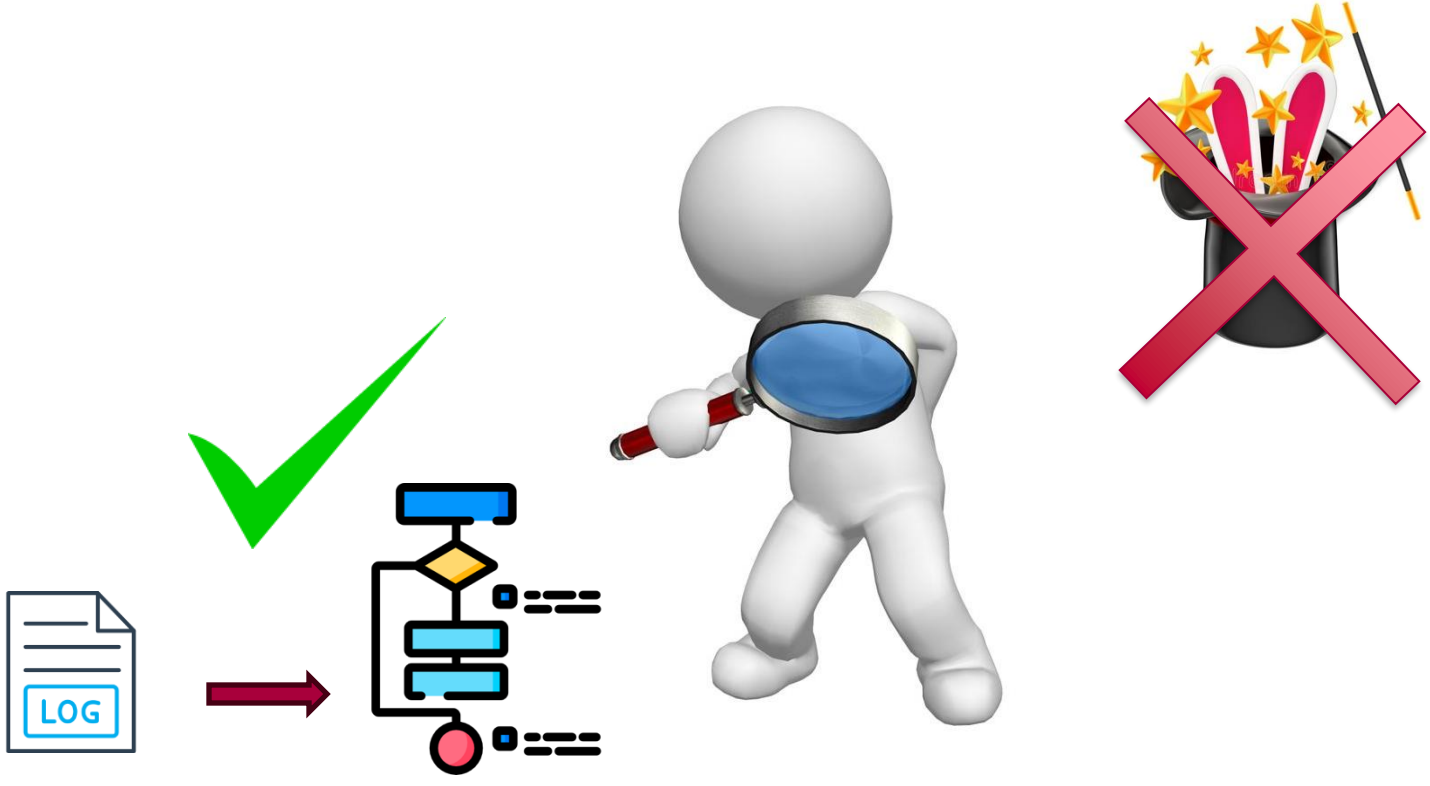


BPMN (Business Process Model and Notation)



BPMN (Business Process Model and Notation)

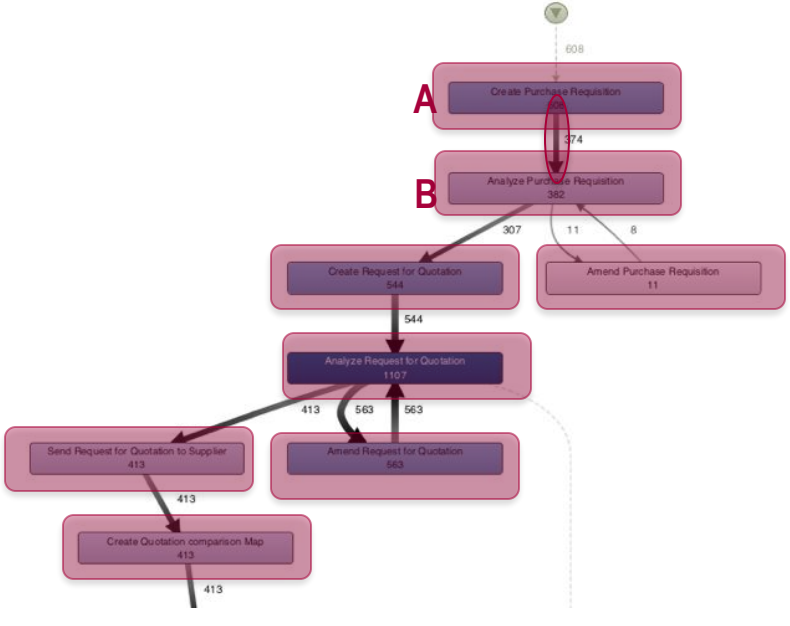



LOG

Mapa de Proceso

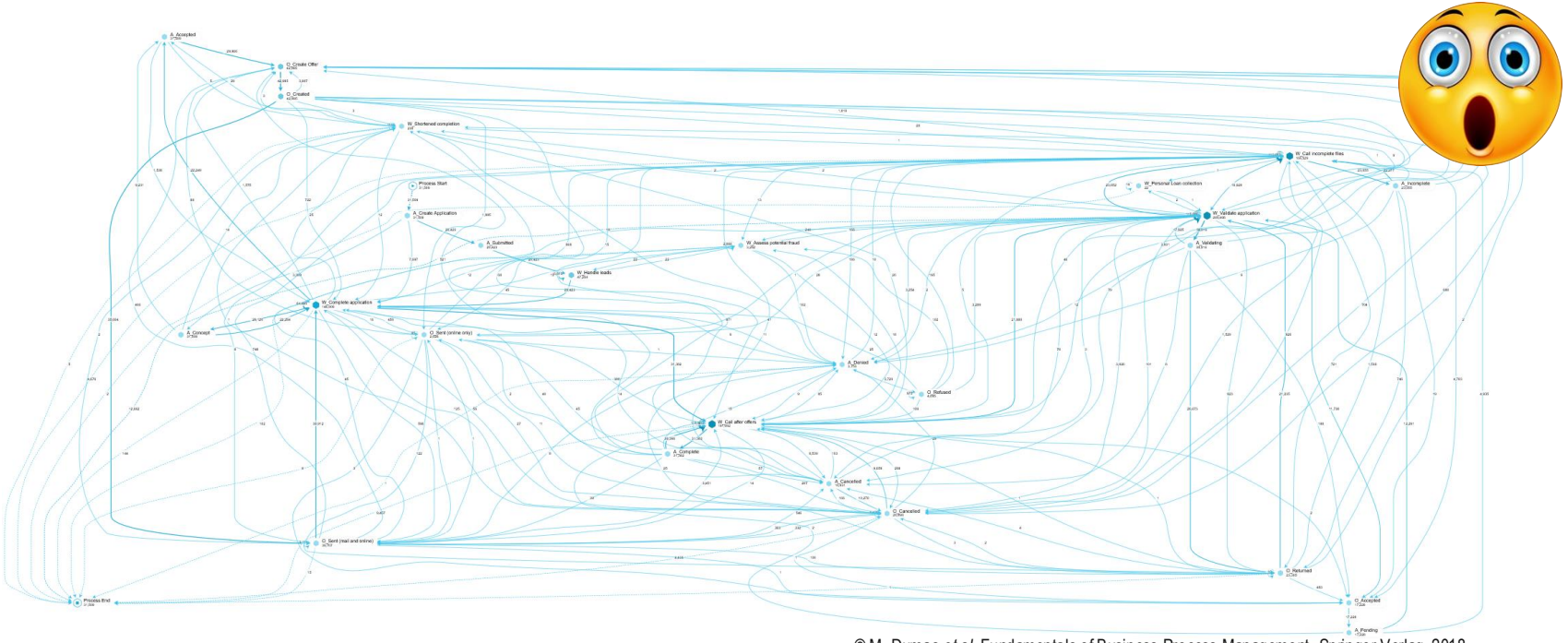
Un **mapa de procesos** (grafo de dependencia o directly-follows graph) de un registro de eventos es un **grafo** donde:

- Cada actividad está representada por un **nodo**
- Un **arco** que va de la actividad A a la actividad B significa que A **es seguido directamente** por B en **al menos una traza del registro**



Operaciones sobre registros de eventos - Ejemplo

Mapa de proceso en bruto de un registro de eventos real (En Celonis)

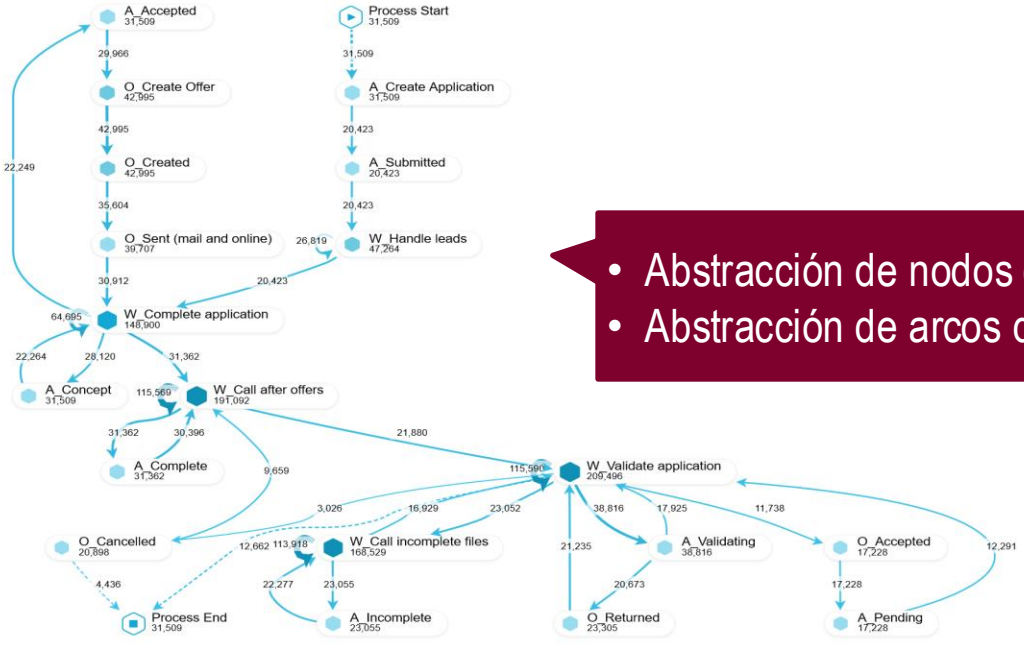


© M. Dumas *et al*. Fundamentals of Business Process Management, Springer-Verlag, 2018

Típicamente se conoce como **"modelo espagueti"**. Para evitar estas salidas, es importante abstraer, filtrar y garantizar una buena calidad de los datos en los registros.

Operaciones sobre registros de eventos - Ejemplo

Mapa de proceso abstraído (En Celonis)



- Abstracción de nodos del 90%
- Abstracción de arcos del 98%

Limitaciones de los Mapas de Procesos

- **Los mapas de proceso sobre-generalizan:** algunos caminos en el mapa de proceso son ficticios.

  Por ejemplo: Dibuja el mapa de proceso del siguiente conjunto de trazas:

  [abc, adc, afce, afec]

  Este mapa de procesos **contiene caminos que no existen** en el registro de eventos

- En los mapas de procesos, suele ser **difícil distinguir entre paralelismo y ciclos**

  Por ejemplo: Dibuja el mapa de procesos del siguiente conjunto de trazas:

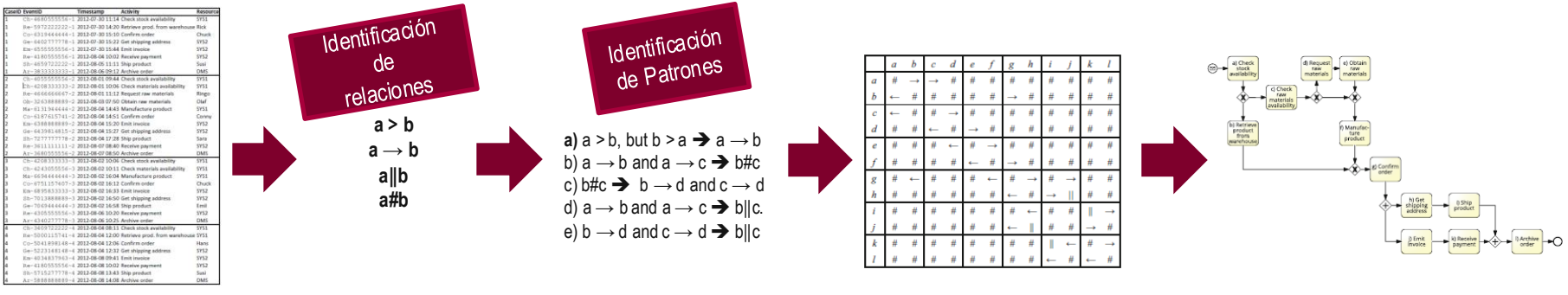
  [abcd, acbd]

- **Solución**: Descubrimiento de modelos de procesos BPMN

Algoritmos de Descubrimiento

Algoritmo Alfa (α)

- Algoritmo más sencillo: Algoritmo α (o Alpha Miner):
  - Detecta la ejecución concurrente de actividades
  - Relativamente fácil, se pueden demostrar ciertas propiedades
  - No es robusto contra el ruido, registros incompletos o erróneos
  - Por lo tanto, agradable para la ilustración, **pero de uso limitado en la práctica**.

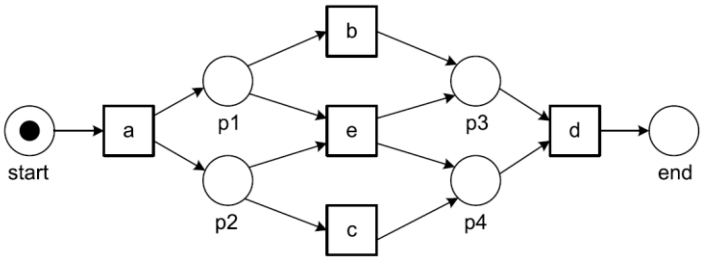


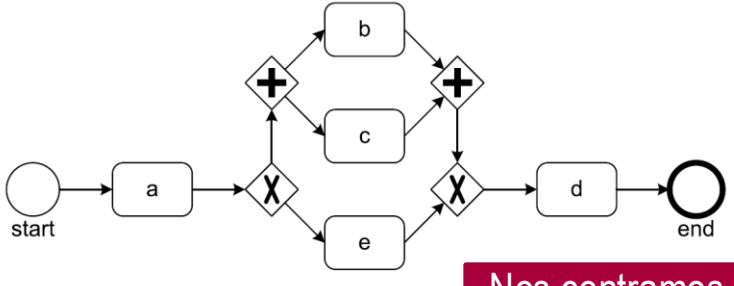
Algoritmo Alfa (α)

Entrada: Log simplificado
Salida: Modelo de proceso

La notación utilizada como salida no es muy relevante. Pueden hacerse transformaciones entre notaciones.

Los tipos de relaciones entre actividades que pueden detectarse y ,por tanto, cómo de expresivo es el modelo obtenido sí son relevantes.





$$L_1 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$$

Nos centramos en actividades

Algoritmo Alfa (α)

Entrada: Log simplificado
El algoritmo alfa construye un modelo de proceso a partir de un registro de eventos **conductualmente completo** (siempre que una tarea a puede ser seguida directamente por una tarea b en el proceso real, hay al menos un caso en el registro de eventos en el que observamos ab)

Las **relaciones de orden** se refieren a pares de tareas que se suceden directamente en el registro.

Relaciones Alfa (α):

- Causalidad.
- Paralelismo potencial.
- No sucesión directa.

Algoritmo Alfa (α): Relaciones

Sucesión directa: a > b

- Se cumple si podemos observar en el event log L que una tarea a va seguida directamente de b.
- Esta es la misma relación que se captura en un gráfico de dependencia.

$$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$$

a>b a>c a>e b>c b>d c>b c>d e>d

A partir de la relación de sucesión directa se pueden descubrir tipos adicionales de relaciones

Algoritmo Alfa (α): Relaciones

Causalidad: a → b

- Se deriva de la relación de sucesión directa.
- Se cumple si observamos en L que a > b y que b ≯ a.

$$L_1 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$$

a>b a>c a>e b>c b>d c>b c>d e>d

a⟶b a⟶c a⟶e b⟶d c⟶d e⟶d

Algoritmo Alfa (α): Relaciones

Paralelismo: a || b

- Se cumple si se observan a > b y b > a en el log de eventos L.

$$L_1 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$$

a>b a>c a>e b>c b>d c>b c>d e>d

a⟶b a⟶c a⟶e b⟶d c⟶d e⟶d

b || c c || b

Algoritmo Alfa (α): Relaciones

Sucesión no directa (elección condicional): a # b

- Se cumple si se observan a ≯ b y b ≯ a en el event log L.

$$L_1 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$$

a>b a>c a>e b>c b>d c>b c>d e>d

a⟶b a⟶c a⟶e b⟶d c⟶d e⟶d

b || c c || b

b#e e#b c#e a#d

Algoritmo Alfa (α)

Footprint: Una representación de las relaciones entre actividades como una matriz de adyacencia anotada.

$$L_1 = \left[\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle\right]$$

Sucesión directa	a>b a>c a>e b>c b>d c>b c>d e>d
Causalidad	a→b a→c a→e b→d c→d e→d
Paralelismo	b || c c || b
Sucesión no directa	b#e e#b c#e a#d

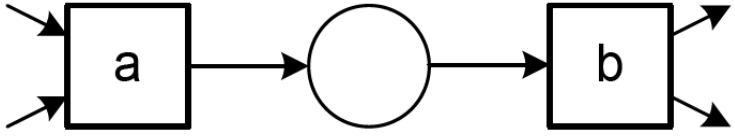
	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

Algoritmo Alfa (α): 5 patrones derivados de las relaciones α

1. *Patrón secuencia: a → b*

Representa una secuencia de tareas *a* y *b*. Se debe garantizar que en un event log encontraremos *a* seguido de *b*, es decir, $a>b$, pero nunca *b* seguido de *a*, es decir, $b \not> a$.

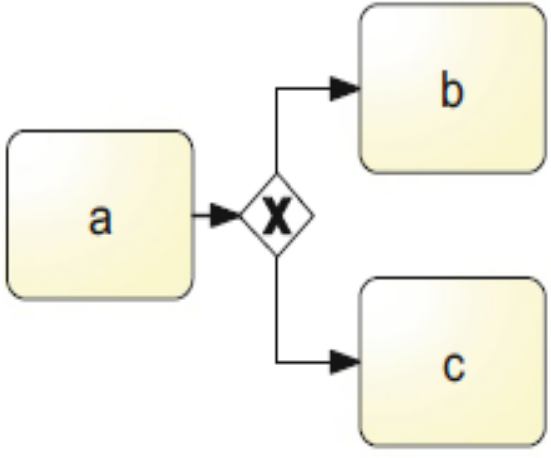
Esto significa que la relación de causalidad $a \rightarrow b$ debe mantenerse.

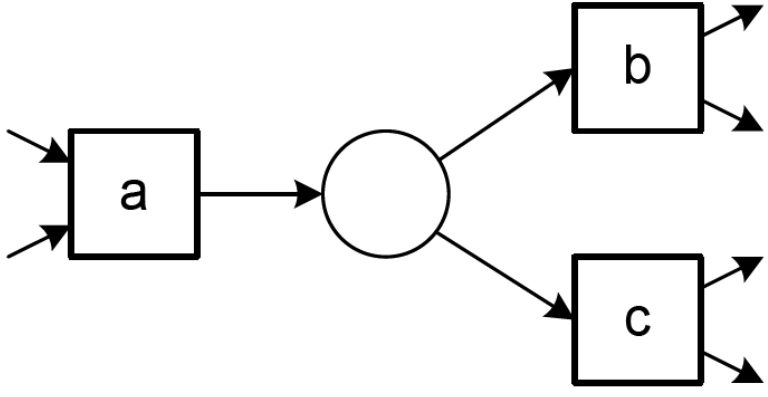


Algoritmo Alfa (α): 5 patrones derivados de las relaciones α

2. *Patrón XOR-split: a → b, a → c, b # c*

El event log debe mostrar que *a→b y a→c* se mantienen, y que *b y c* no serían sucesores mutuos, es decir, b # c.

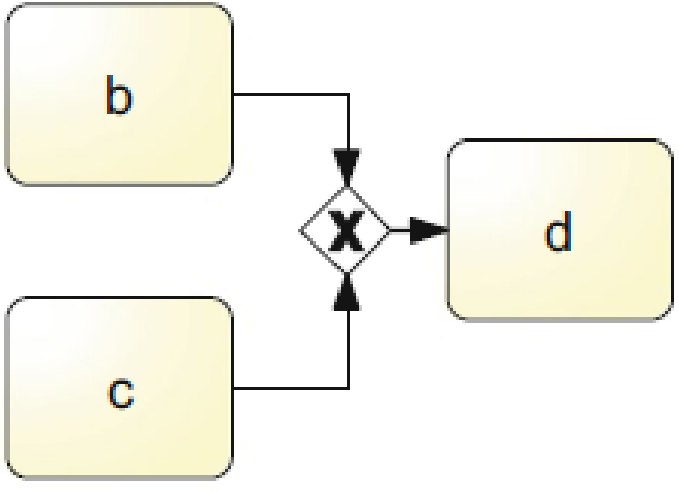


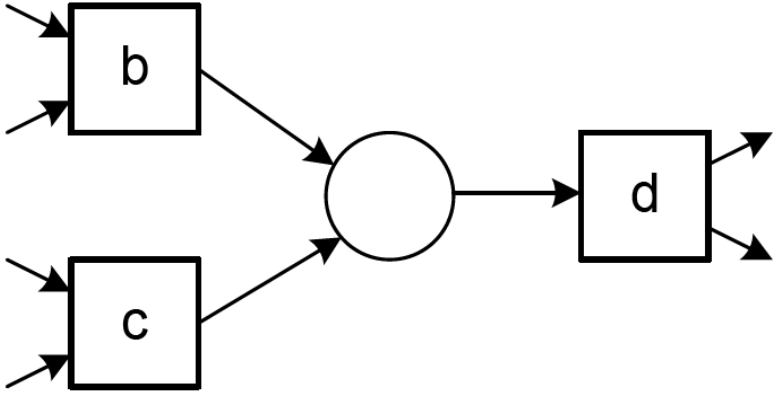


Algoritmo Alfa (α): 5 patrones derivados de las relaciones α

3. *Patrón join: b → d, c → d, b # c*

Requiere que *b y c* no sean sucesores mutuos, es decir, b # c, mientras que *a→b y a→c* deben cumplirse.

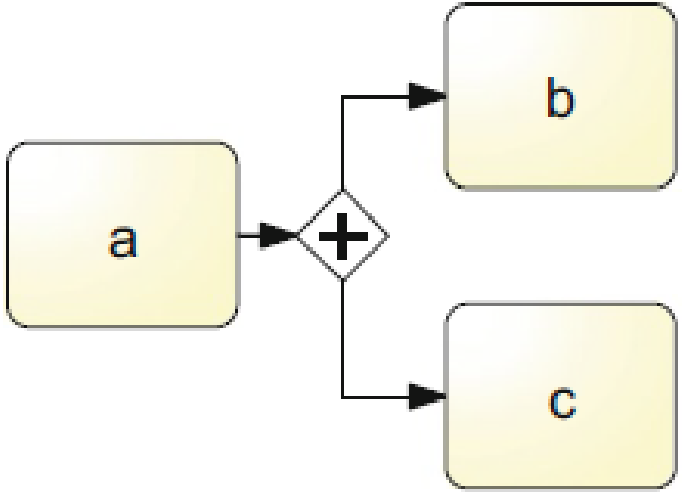


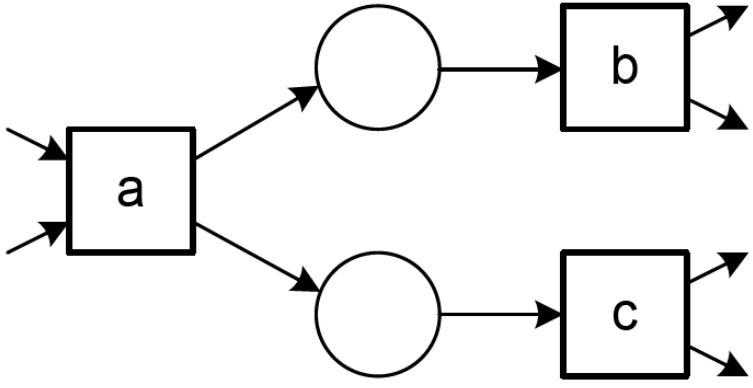


Algoritmo Alfa (α): 5 patrones derivados de las relaciones α

4. *Patrón AND-split: a → b, b → c, b || c*

Exige que a→b y a→c se cumplan, y que b y c muestren un paralelismo potencial, es decir, b||c.

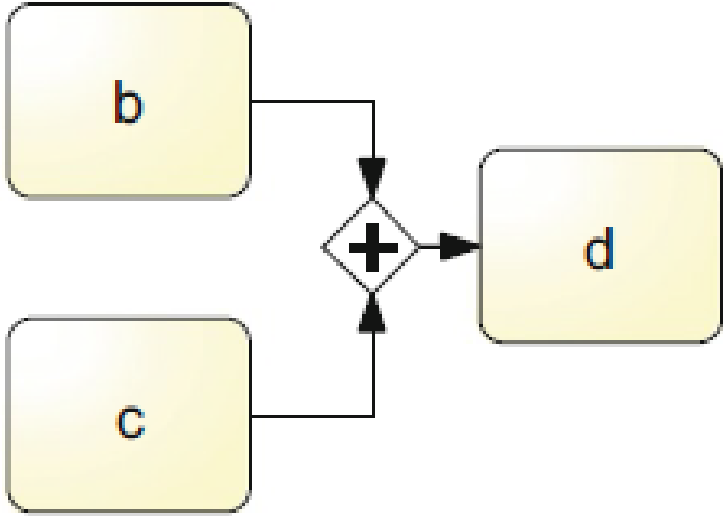


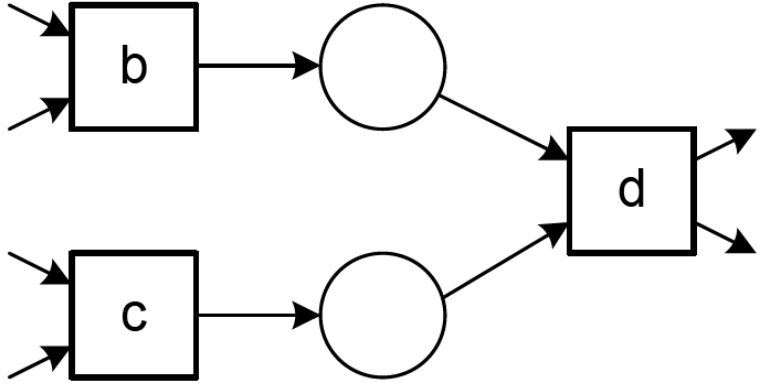


Algoritmo Alfa (α): 5 patrones derivados de las relaciones α

5. *Patrón AND-join: b → d, c → d, b || c*

Se refiere a b→d y c→d, mientras que b y c muestran un paralelismo potencial, es decir, b||c.





Algoritmo Alfa (α): Ejemplo

Considere un registro de eventos con dos trazas:

< a, b, g, h, j, k, i, l > y < a, c, d, e, f, g, j, h, i, k, l>

Sucesión directa:

$a > b$	$h > j$	$i > l$	$d > e$	$g > j$	$i > k$
$b > g$	$j > k$	$a > c$	$e > f$	$j > h$	$k > l$
$g > h$	$k > i$	$c > d$	$f > g$	$h > i$	

Causalidad:

$a \rightarrow b$	$j \rightarrow k$	$a \rightarrow c$	$d \rightarrow e$	$f \rightarrow g$	$h \rightarrow i$
$b \rightarrow g$	$i \rightarrow l$	$c \rightarrow d$	$e \rightarrow f$	$g \rightarrow j$	$k \rightarrow l$
$g \rightarrow h$					

Paralelismo potencial es válida tanto para *h* || *j* como para k || i (y los casos simétricos correspondientes).

Sucesión no directa se puede encontrar para todos los pares que no pertenecen a la causalidad → y el paralelismo ||

Algoritmo Alfa (α): Ejemplo

Footprint: Una representación de las relaciones entre actividades como una matriz de adyacencia anotada.

Causalidad:

$a \rightarrow b$	$j \rightarrow k$	$a \rightarrow c$	$d \rightarrow e$	$f \rightarrow g$	$h \rightarrow i$
$b \rightarrow g$	$i \rightarrow l$	$c \rightarrow d$	$e \rightarrow f$	$g \rightarrow j$	$k \rightarrow l$
$g \rightarrow h$					

Paralelismo potencial:

$h \parallel j$
$k \parallel i$

	a	b	c	d	e	f	g	h	i	j	k	l
a	#	$\rightarrow$	$\rightarrow$	#	#	#	#	#	#	#	#	#
b	$\leftarrow$	#	#	#	#	#	$\rightarrow$	#	#	#	#	#
c	$\leftarrow$	#	#	$\rightarrow$	#	#	#	#	#	#	#	#
d	#	#	$\leftarrow$	#	$\rightarrow$	#	#	#	#	#	#	#
e	#	#	#	$\leftarrow$	#	$\rightarrow$	#	#	#	#	#	#
f	#	#	#	#	$\leftarrow$	#	$\rightarrow$	#	#	#	#	#
g	#	$\leftarrow$	#	#	#	$\leftarrow$	#	$\rightarrow$	#	$\rightarrow$	#	#
h	#	#	#	#	#	#	$\leftarrow$	#	$\rightarrow$	$\parallel$	#	#
i	#	#	#	#	#	#	#	$\leftarrow$	#	#	$\parallel$	$\rightarrow$
j	#	#	#	#	#	#	$\leftarrow$	$\parallel$	#	#	$\rightarrow$	#
k	#	#	#	#	#	#	#	#	$\parallel$	$\leftarrow$	#	$\rightarrow$
l	#	#	#	#	#	#	#	#	$\leftarrow$	#	$\leftarrow$	#

Algoritmo Alfa (α): Ejemplo

Hasta ahora hemos identificado las relaciones de orden, pero ¡esto no es el Algoritmo Alfa!

Buscamos **obtener un modelo de proceso**

Partimos de:

- El registro de eventos:
  $L = [< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$
- La matriz – Footprint
- Aplicamos un algoritmo de 8 pasos.

Algoritmo Alfa (α): Ejemplo

1.- Sea T_L el conjunto de todas las tareas en el registro.

2.- Sea T_I el conjunto de tareas que aparecen al menos una vez como primera tarea de un caso.

3.- Sea T_O el conjunto de tareas que aparecen al menos una vez como última tarea de un caso.

4.- Sea X_L el conjunto de posibles conexiones de tareas. XL se compone de:

- Patrón (1): todos los pares para los que se cumple a → b.
- Patrón (2): todos los triples para los que se cumple a → (b#c).
- Patrón (3): todos los triples para los cuales se cumple (b#c) → d.

Tenga en cuenta que los triples para los cuales se mantiene el Patrón (4) a → (b||c) o el Patrón (5) (b||c) → d no están incluidos en X_L.

5.- Construya el conjunto YL como un subconjunto de XL mediante:

- Eliminando a → b y a → c si existe algún a → (b#c).
- Eliminando b → c y b → d si existe algún (b#c) → d.

6.- Conecte los eventos de inicio y fin.

7.- Construya los arcos de flujo de la siguiente manera:

- Patrón (1): para cada a→b en YL, dibuje un arco de a a b.
- Patrón (2): para cada a→(b#c) en YL, dibuje un arco desde a hasta una división XOR (XOR-split), y desde allí hasta b y c.
- Patrón (3): para cada (b#c)→d en YL, dibuje un arco desde b y c hasta una unión XOR, y desde allí hasta d.
- Patrón (4) y (5): si una tarea en el modelo de proceso así construido tiene múltiples arcos entrantes o múltiples salientes, agrupe estos arcos con una unión AND o una división AND, respectivamente.

8. Devolver el proceso recién construido.

Algoritmo Alfa (α): Ejemplo – 8 pasos

Log: $[< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$

1.- Sea T_L el conjunto de todas las tareas en el registro.

$$T_L = \{a, b, c, d, e, f, g, h, i, j, k, l\}$$

2.- Sea T_I el conjunto de tareas que aparecen al menos una vez como primera tarea de un caso.

$$T_I = \{ a \}$$

3.- Sea T_O el conjunto de tareas que aparecen al menos una vez como última tarea de un caso.

$$T_O = \{ l \}$$

Algoritmo Alfa (α): Ejemplo – 8 pasos

Log: $[< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$

4.- Sea X_L el conjunto de posibles conexiones de tareas. X_L se compone de:

- Patrón (1): todos los pares para los que se cumple a ⟶ b. Causalidad
- Patrón (2): todos los triples para los que se cumple a ⟶ (b#c). Usa la matriz
- Patrón (3): todos los triples para los cuales se cumple (b#c) ⟶ d.

Tenga en cuenta que los triples para los cuales se mantiene el Patrón (4) a⟶(b||c) o el Patrón (5) (b||c) ⟶ d no están incluidos en X_L.

$\boldsymbol{X_L}$ = (1) U (2) U (3), donde

(1) = {*a⟶b, b⟶g, g⟶h, j⟶k, i⟶l, a⟶c, c⟶d, d⟶e, e⟶f, f⟶g, g⟶j, h⟶i, k⟶l* }

Algoritmo Alfa (α): Ejemplo – 8 pasos

Log: $[< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$

4.- Sea XL el conjunto de posibles conexiones de tareas. XL se compone de:

- Patrón (1): todos los pares para los que se cumple a ⟶ b. **Causalidad**
- Patrón (2): tc… …c). **Usa la matriz**
- Patrón (3): tc… ⟶ d.

Tenga en cuer… …el Patrón (4) a⟶(b||c) o el Patrón (5) (b||…

X_L = (1) U (…

(1) = {a⟶b… …f, f⟶g, g⟶j, h⟶i, k⟶l }

	a	b	c	d	e	f	g	h	i	j	k	l
a	#	→	→	#	#	#	#	#	#	#	#	#
b	←	#	#	#	#	#	→	#	#	#	#	#
c	←	#	#	→	#	#	#	#	#	#	#	#
d	#	#	←	#	→	#	#	#	#	#	#	#
e	#	#	#	←	#	→	#	#	#	#	#	#
f	#	#	#	#	←	#	→	#	#	#	#	#
g	#	←	#	#	#	←	#	→	#	→	#	#
h	#	#	#	#	#	#	←	#	→	||	#	#
i	#	#	#	#	#	#	#	←	#	#	||	→
j	#	#	#	#	#	#	←	||	#	#	→	#
k	#	#	#	#	#	#	#	#	||	←	#	→
l	#	#	#	#	#	#	#	#	←	#	←	#

Algoritmo Alfa (α): Ejemplo – 8 pasos

Log: $[< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$

4.- Sea X_L el conjunto de posibles conexiones de tareas. X_L se compone de:

- Patrón (1): todos los pares para los que se cumple a ⟶ b. Causalidad
- Patrón (2): todos los triples para los que se cumple a ⟶ (b#c). Usa la matriz
- Patrón (3): todos los triples para los cuales se cumple (b#c) ⟶ d. Usa la matriz

Tenga en cuenta que los triples para los cuales se mantiene el Patrón (4) a⟶(b||c) o el Patrón (5) (b||c) ⟶ d no están incluidos en X_L.

$\boldsymbol{X_L}$ = (1) U (2) U (3), donde

(1) = {a⟶b, b⟶g, g⟶h, j⟶k, i⟶l, a⟶c, c⟶d, d⟶e, e⟶f, f⟶g, g⟶j, h⟶i, k⟶l }

(2) = {a⟶ (b # c)}

Algoritmo Alfa (α): Ejemplo – 8 pasos

Log: $[< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$

4.- Sea X_L el conjunto de posibles conexiones de tareas. X_L se compone de:

- Patrón (1): todos los pares para los que se cumple a ⟶ b. Causalidad
- Patrón (2): todos lo… sa la matriz
- Patrón (3): todos lo… Usa la matriz

Tenga en cuenta que… ón (4) a⟶(b||c) o el Patrón (5) (b||c) ⟶ c…

	a	b	c	d	e	f	g	h	i	j	k	l
a	#	→	→	#	#	#	#	#	#	#	#	#
b	←	#	#	#	#	#	→	#	#	#	#	#
c	←	#	#	→	#	#	#	#	#	#	#	#
d	#	#	←	#	→	#	#	#	#	#	#	#
e	#	#	#	←	#	→	#	#	#	#	#	#
f	#	#	#	#	←	#	→	#	#	#	#	#
g	#	←	#	#	#	←	#	→	#	→	#	#
h	#	#	#	#	#	#	←	#	→	||	#	#
i	#	#	#	#	#	#	#	←	#	#	||	→
j	#	#	#	#	#	#	←	||	#	#	→	#
k	#	#	#	#	#	#	#	#	||	←	#	→
l	#	#	#	#	#	#	#	#	←	#	←	#

X_L = (1) U (2) U …

(1) = {a⟶b, b⟶g… g, g⟶j, h⟶i, k⟶l }

(2) = {a⟶ (b # c …

Algoritmo Alfa (α): Ejemplo – 8 pasos

Log: $[< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$

4.- Sea X_L el conjunto de posibles conexiones de tareas. X_L se compone de:

- Patrón (1): todos los pares para los que se cumple a ⟶ b. Causalidad
- Patrón (2): todos los triples para los que se cumple a ⟶ (b#c). Usa la matriz
- Patrón (3): todos los triples para los cuales se cumple (b#c) ⟶ d. Usa la matriz

Tenga en cuenta que los triples para los cuales se mantiene el Patrón (4) a⟶(b||c) o el Patrón (5) (b||c) ⟶ d no están incluidos en X_L.

X_L = (1) U (2) U (3), donde

(1) = {a⟶b, b⟶g, g⟶h, j⟶k, i⟶l, a⟶c, c⟶d, d⟶e, e⟶f, f⟶g, g⟶j, h⟶i, k⟶l }
(2) = {a⟶ (b # c)}
(3) = {(b # f) ⟶ g}

No incluimos (i || k) ⟶ g

Algoritmo Alfa (α): Ejemplo – 8 pasos

Log: $[< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$

5.- Construya el conjunto Y_L como un subconjunto de X_L mediante:

- Eliminando a ⟶ b y a ⟶ c si existe algún a ⟶ (b#c).
- Eliminando b ⟶ c y b ⟶ d si existe algún (b#c) ⟶ d.

$\boldsymbol{X_L}$ = (1) U (2) U (3), donde

(1) = {~~a⟶b~~, ~~b⟶g~~, g⟶h, j⟶k, i⟶l, ~~a⟶c~~, c⟶d, d⟶e, e⟶f, ~~f⟶g~~, g⟶j, h⟶i, k⟶l }

(2) = {a⟶ (b # c)}

(3) = {(b # f) ⟶ g}

$\boldsymbol{Y_L}$ = (1) U (2) U (3), donde

(1) = {g⟶h, j⟶k, i⟶l, c⟶d, d⟶e, e⟶f, g⟶j, h⟶i, k⟶l }

(2) = {a⟶ (b # c) }

(3) = {(b # f) ⟶ g}

Algoritmo Alfa (α): Ejemplo – 8 pasos

Log: $[< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$

6.- Conecte los eventos de inicio y fin de la siguiente manera:

- Si hay varias tareas en el conjunto T_I de las primeras tareas, dibuje un evento de inicio que conduzca a una división XOR, que se conecta a cada tarea en T_I. De lo contrario, conecte directamente el evento de inicio con la primera tarea.
- Para cada tarea en el conjunto T_O de las últimas tareas, agregue un evento final y dibuje un arco desde la tarea hasta el evento final.

¡Empezamos a modelar!

Algoritmo Alfa (α): Ejemplo – 8 pasos

Log: $[< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$

7.- Construya los arcos de flujo de la siguiente manera:

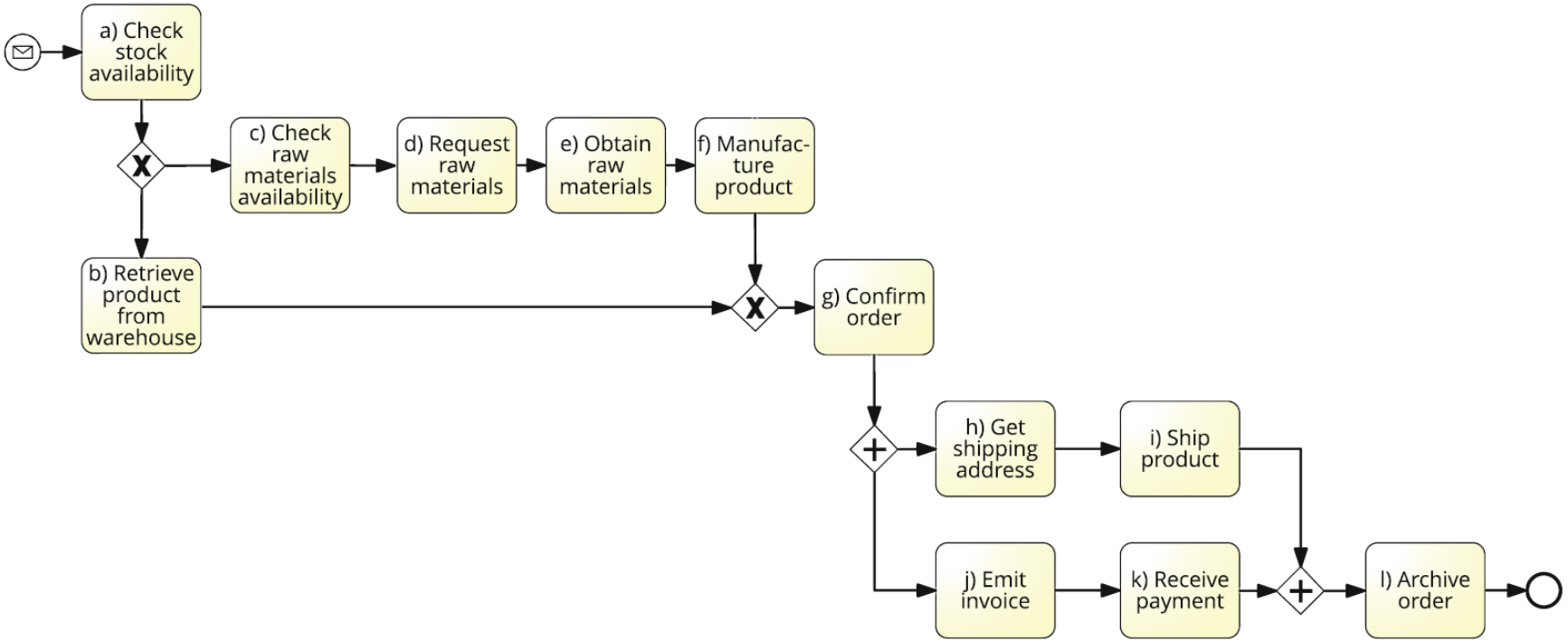
- Patrón (1): para cada $a \rightarrow b$ en Y_L, dibuje un arco de a a b.
- Patrón (2): para cada $a \rightarrow (b\#c)$ en Y_L, dibuje un arco desde a hasta una *división XOR* (*XOR-split*), y desde allí hasta b y c.
- Patrón (3): para cada $(b\#c) \rightarrow d$ en Y_L, dibuje un arco desde b y c hasta una *unión XOR*, y desde allí hasta d.
- Patrón (4) y (5): si una tarea en el modelo de proceso así construido tiene múltiples arcos entrantes o múltiples salientes, agrupe estos arcos con una *unión AND* o una *división AND*, respectivamente.

8.- Devolver el proceso recién construido.

Algoritmo Alfa (α): Ejemplo – 8 pasos

Modelo obtenido por el Algoritmo α para el registro:

$$L = [< a, b, g, h, j, k, i, l >, < a, c, d, e, f, g, j, h, i, k, l >]$$



Fundamentals of Business Process Management. Springer Berlin, Heidelberg, 2018.

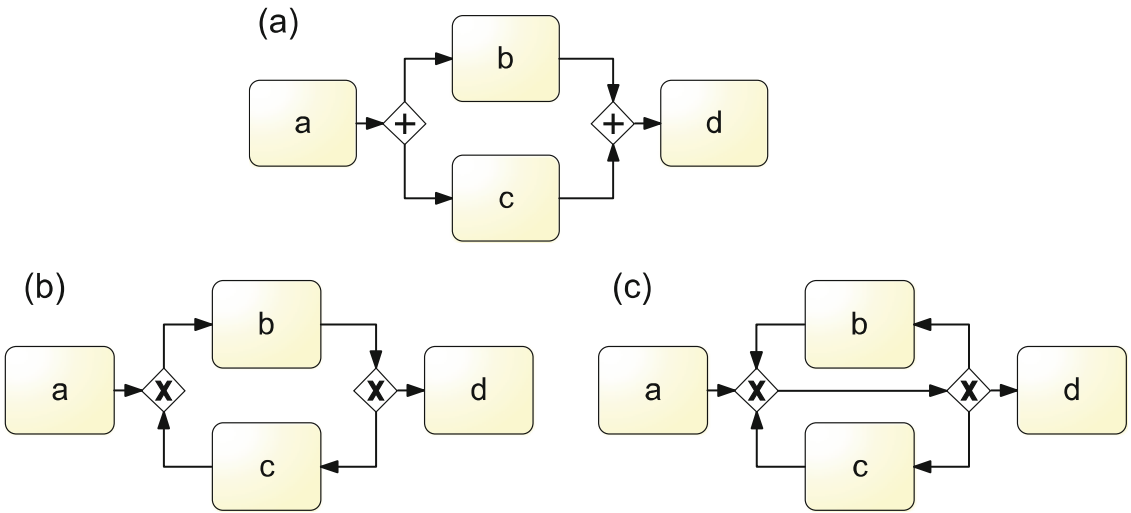
Algoritmo Alfa (α): Limitaciones

No es robusto:

- **No es capaz de detectar bucles cortos**

Por ejemplo:

El Algoritmo α no es capaz de distinguir entre estos 3 fragmentos. b y c son bucles cortos y los define siempre como el fragmento a.



Fundamentals of Business Process Management. Springer Berlin, Heidelberg, 2018.

Algoritmo Alfa (α): Limitaciones

- **No es capaz de lidiar con datos incompletos en un log de eventos.**

- Supone que la relación > está completa:
  si el proceso permite que la tarea a sea seguida directamente por la tarea b, entonces la relación $a > b$ debe observarse en al menos una traza del registro de eventos.

- Esta suposición es demasiado fuerte para procesos que tienen mucho paralelismo.

Es necesario inferir inteligentemente relaciones que no se observan explícitamente

Por ejemplo:
Si en un proceso hay un bloque de diez tareas concurrentes $a_1, \ldots a_{10}$, necesitamos observar cada relación $a_i > a_j$ para cada $i, j \in [1..10] = 100$ combinaciones.

Basta que una de estas combinaciones no se haya observado en el registro de eventos para que el algoritmo α descubra el modelo incorrecto.

Fundamentals of Business Process Management. Springer Berlin, Heidelberg, 2018.

Algoritmo Alfa (α): Limitaciones

- **No es capaz de lidiar con el ruido.**

- Los registros de eventos a menudo incluyen casos a los que les falta un inicio, un final o un episodio intermedio faltante porque algunos eventos de un caso no se han registrado.

- Puede haber errores de registro que provoquen que los eventos se registren en el orden incorrecto (o con marcas de tiempo incorrectas) o se registren dos veces.

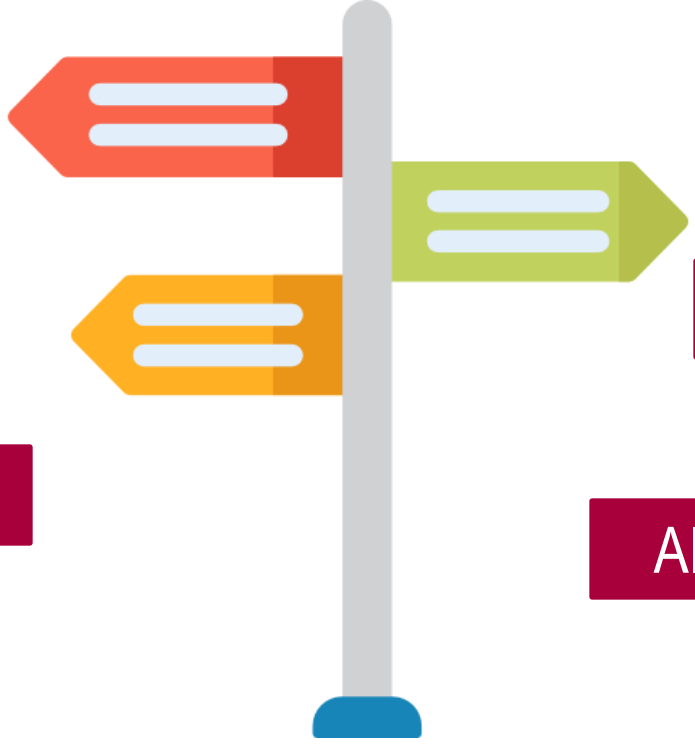
Esto no debería afectar el modelo del proceso descubierto

Alternativas al Algoritmo Alfa (α)

Algoritmo Heurístico

Algoritmo Inductivo

Algoritmo α++



Algoritmo Estructurado

Algoritmo α+

Algoritmo Split Miner

Algoritmos de Descubrimiento

La Familia de los Algoritmos Alfa (α)

- Extensiones: Algoritmos α+ (2003) y α++ (2007):
  - α+ y α++ extienden el algoritmo α para ampliar el espectro de construcciones que pueden ser descubiertas
  - Siguen sin ser robustos frente al ruido
  - Otros derivados posteriores:
    - α# (2010)
    - $\alpha$$ (2015)

Algoritmos de Descubrimiento

Algoritmo Heuristic Miner

Algoritmo Structured Heuristic Miner

Algoritmo Inductive Miner

Algoritmo Split Miner

- **Generalmente funcionan como sigue:**

1. Construyen el gráfico de dependencia a partir del registro de eventos.
2. Eliminan algunos de los nodos y arcos en el gráfico de dependencia del proceso para abordar el ruido.
3. Aplican un conjunto de reglas heurísticas para descubrir compuertas de enlace (de división y unión) para convertir el gráfico de dependencia filtrado en un modelo de proceso.

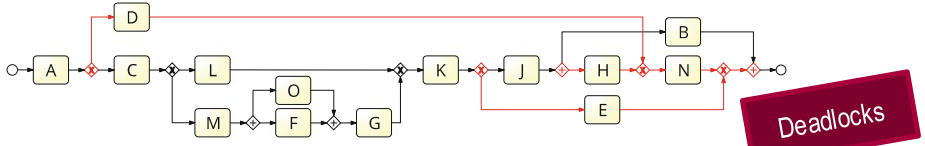
Algoritmos de Descubrimiento

Algoritmos Más Avanzados

- Algoritmo Heurístico (o **Heuristics Miner**):
  - Robusto ante el ruido, rápido, a menudo un buen compromiso, pero puede producir modelos incorrectos.
- Algoritmo Inductivo (**Inductive Miner**)
  - Garantiza que los modelos de proceso generados estén **estructurados en bloques** y sean correctos
  - Por ejemplo, en ProM v6
- Algoritmo Estructurado (**Structured Miner**)
  - Mejora el algoritmo heurístico para producir modelos de proceso con la máxima estructura de bloques
  - Por ejemplo, en Apromore

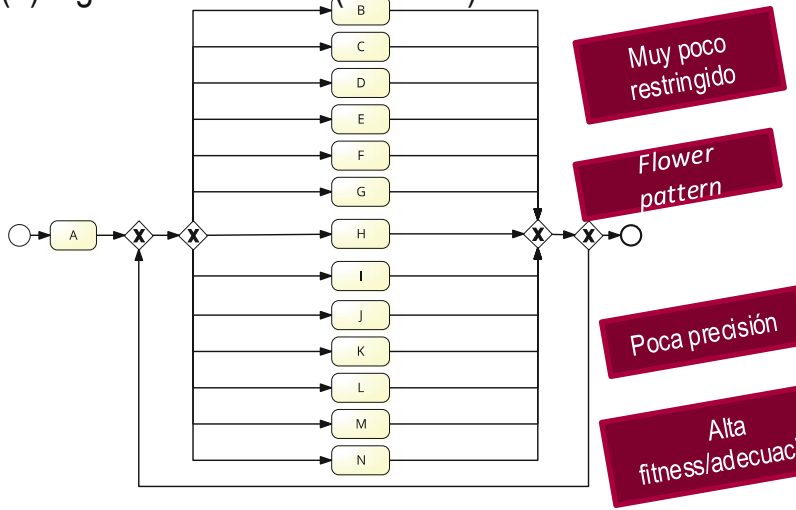
Modelos descubiertos por distintos algoritmos a partir del mismo registro de eventos

(a) Algoritmo heurístico (ProM v6)



Deadlocks

(b) Algoritmo inductivo (ProM v6)



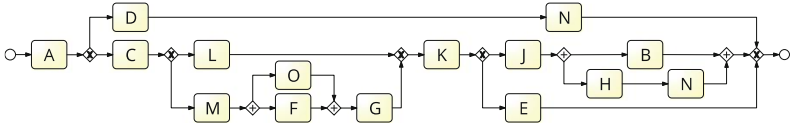
Muy poco restringido

Flower pattern

Poca precisión

Alta fitness/adecuación

(c) Algoritmo heurístico estructurado (Apromore)



Estructurado

Actividades repetidas (N)

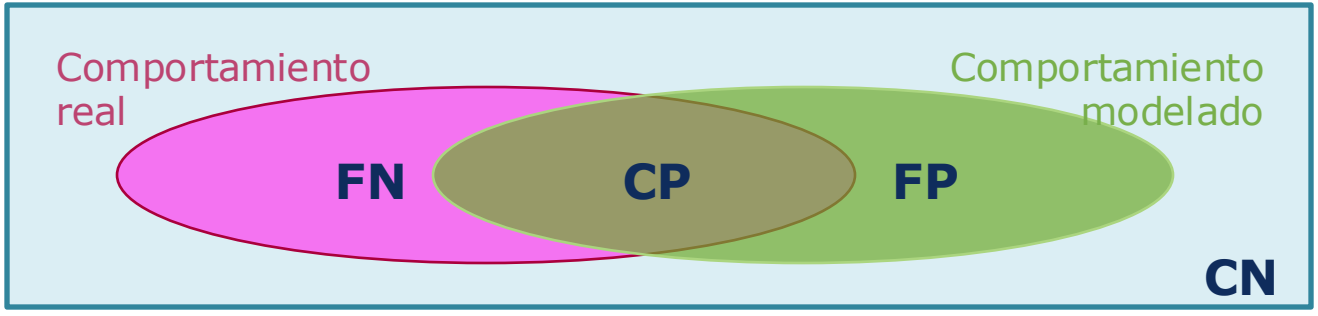
Características del descubrimiento automático de procesos

Fitness (Adecuación)	Es la capacidad del modelo de proceso descubierto para reproducir el comportamiento contenido en el registro.
Precisión	Se refiere a la medida en que el modelo de proceso descubierto genera sólo las trazas encontradas en el registro.
Generalización	Se refiere al grado en que el modelo de proceso descubierto captura trazas que no están presentes en el registro (incompleto), pero que probablemente sean permitidos por el proceso subyacente.
Complexity	Puede ser medida en términos de tamaño (número de nodos o número de flujos en el modelo dividido entre el número de nodos).

Características del descubrimiento automático de procesos

	Adecuación/ Fitness	Precisión	Complejidad
Heuristic Minner	ALTA	ALTA	-
Inductive Minner	ALTA	-	BAJA
Structured Minner	ALTA	ALTA	BAJA
Split Minner	ALTA	ALTA	BAJA
Fodina Minner	ALTA	ALTA	-
Evolutionary Tree Minner	ALTA	-	BAJA

Aproximación a conceptos de clasificación

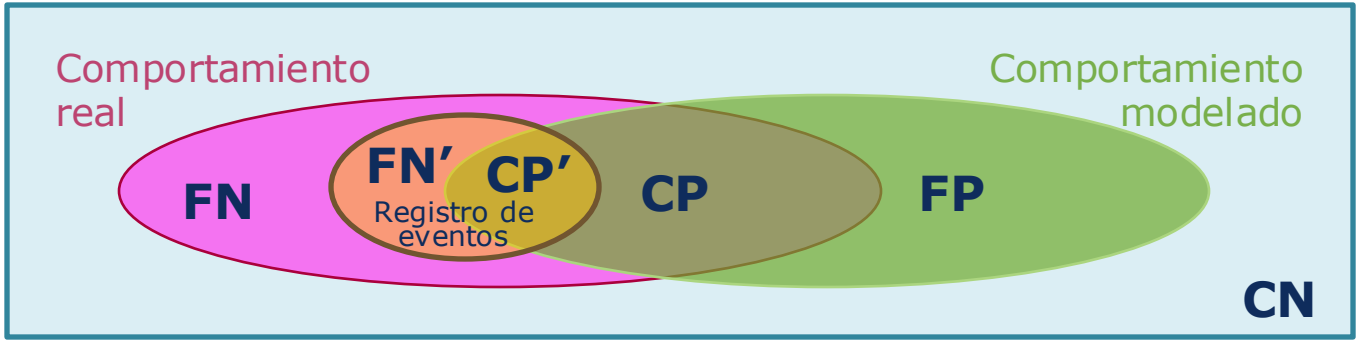


CP Ciertos positivos	Trazas que existen en la realidad y que se pueden reproducir con el modelo.
CN Ciertos negativos	Trazas que no existen en la realidad y que no se pueden reproducir con el modelo.
FP Falsos positivos	Trazas que no existen en la realidad y que se pueden reproducir con el modelo.
FN Falsos negativos	Trazas que existen en la realidad y que no pueden reproducirse con el modelo.

$$recall = \frac{CP}{CP + FN}$$

$$precision = \frac{CP}{CP + FP}$$

Aproximación a conceptos de clasificación



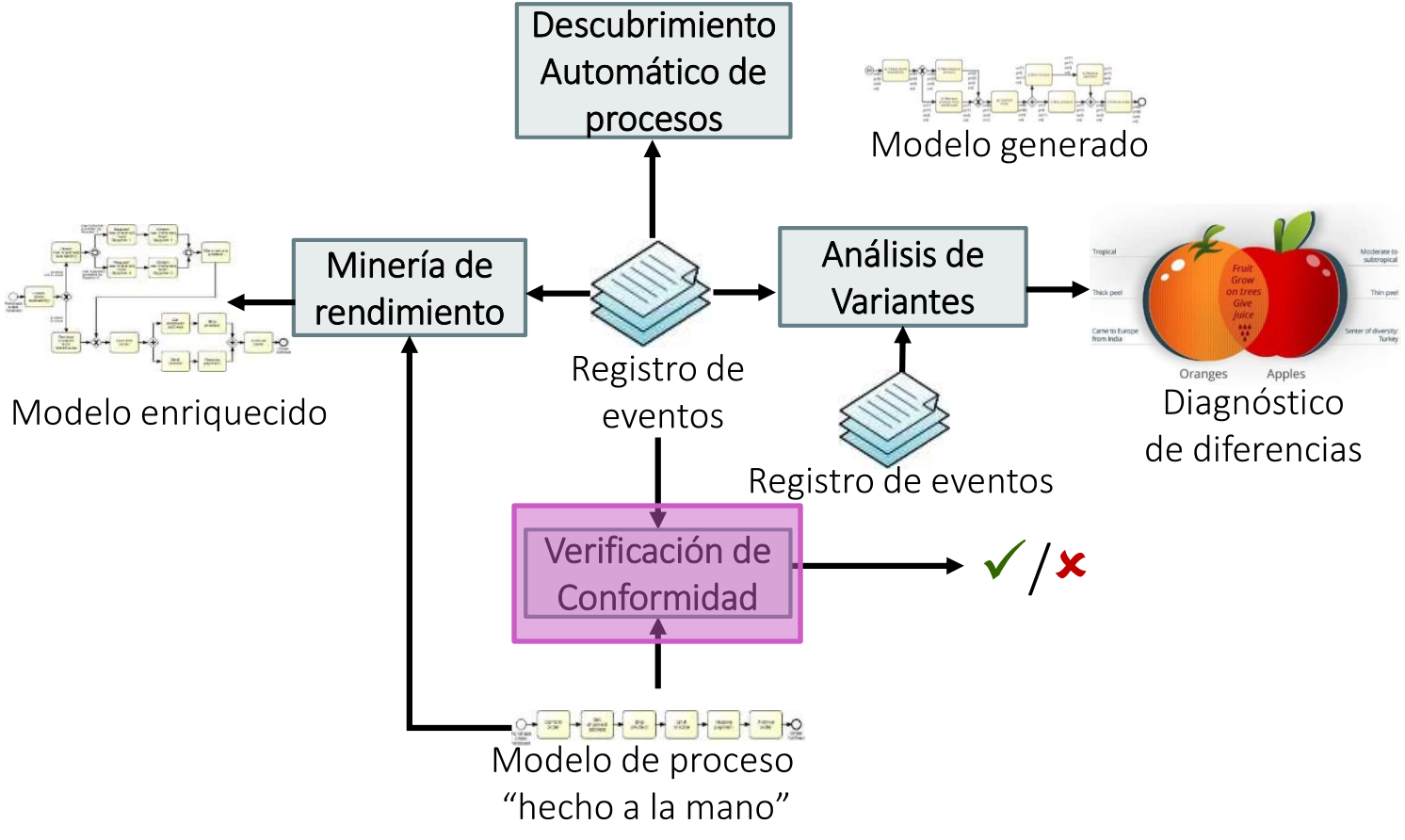
El log de eventos contiene típicamente una fracción de trazas posibles

CP Ciertos positivos	Trazas que existen en la realidad y que se pueden reproducir con el modelo.
FN Falsos negativos	Trazas que existen en la realidad y que no pueden reproducirse con el modelo.

$$replay\_fitness = \frac{CP'}{CP' + FP'}$$

Descubrimiento automático de procesos
Verificación de conformidad

Casos de uso


Descubrimiento Automático de procesos
Modelo generado
Minería de rendimiento
Modelo enriquecido
Registro de eventos
Análisis de Variantes
Diagnóstico de diferencias
Registro de eventos
Verificación de Conformidad
✓ / ✗
Modelo de proceso "hecho a la mano"

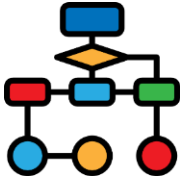
Verificación de conformidad

Si una restricción particular no se cumple, hablamos de una **violación**.

La verificación de conformidad se ocupa de identificar estas violaciones y hacer declaraciones sobre el alcance de las mismas.

Las violaciones pueden estar relacionadas con tres perspectivas de proceso

Flujo de control



Datos

Recursos

Conformidad de flujo de control

Restricciones de flujo de control

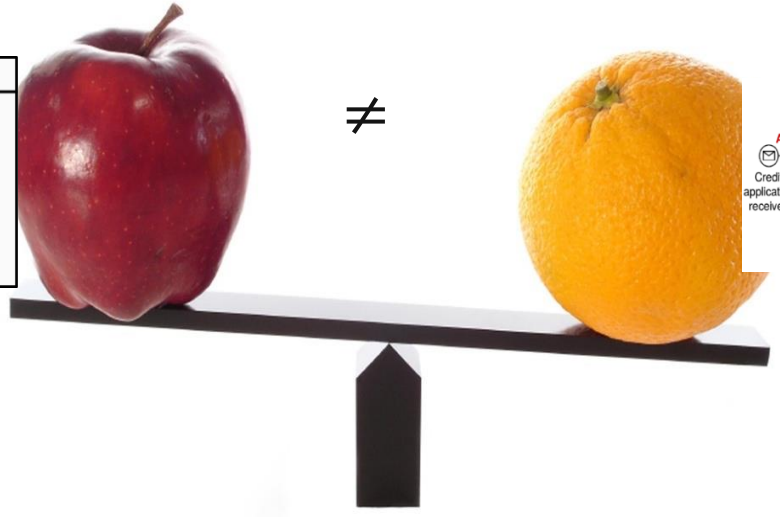
Definen cómo se permite relacionar dos tareas en un proceso.

- Obligatoriedad
- Exclusividad
- Ordenación

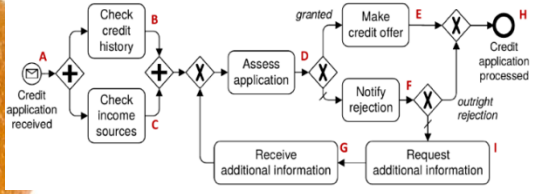
Causalidad	a→b
Paralelismo	b || c
Sucesión no directa	b#e

Conformidad de flujo de control

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

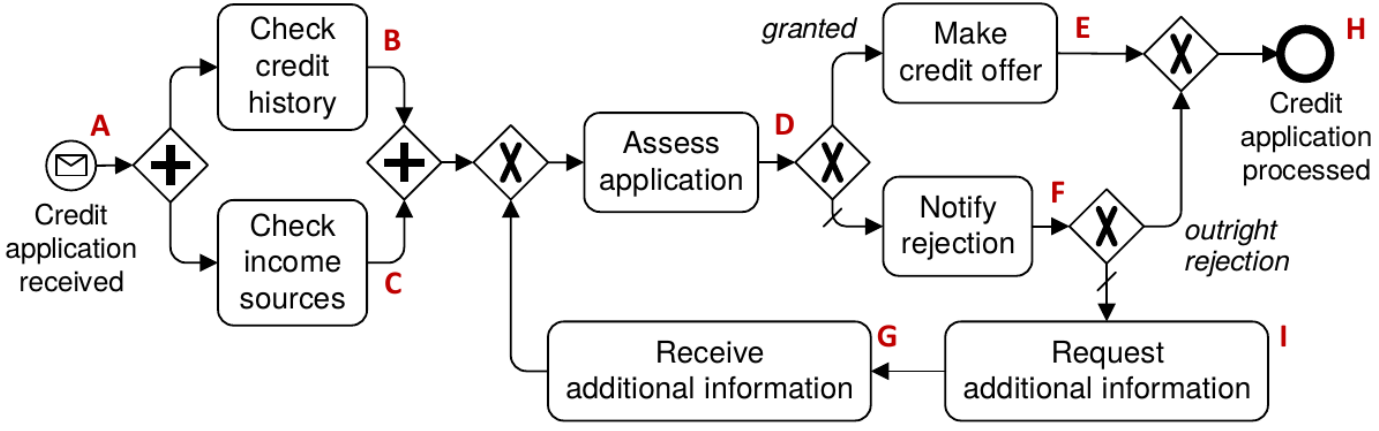


≠



¿Coincide la información del registro de eventos con el modelo de proceso predefinido?

Para practicar



Event log:

ABCDEH

ACBDEH

ABCDFH

ACBDFH

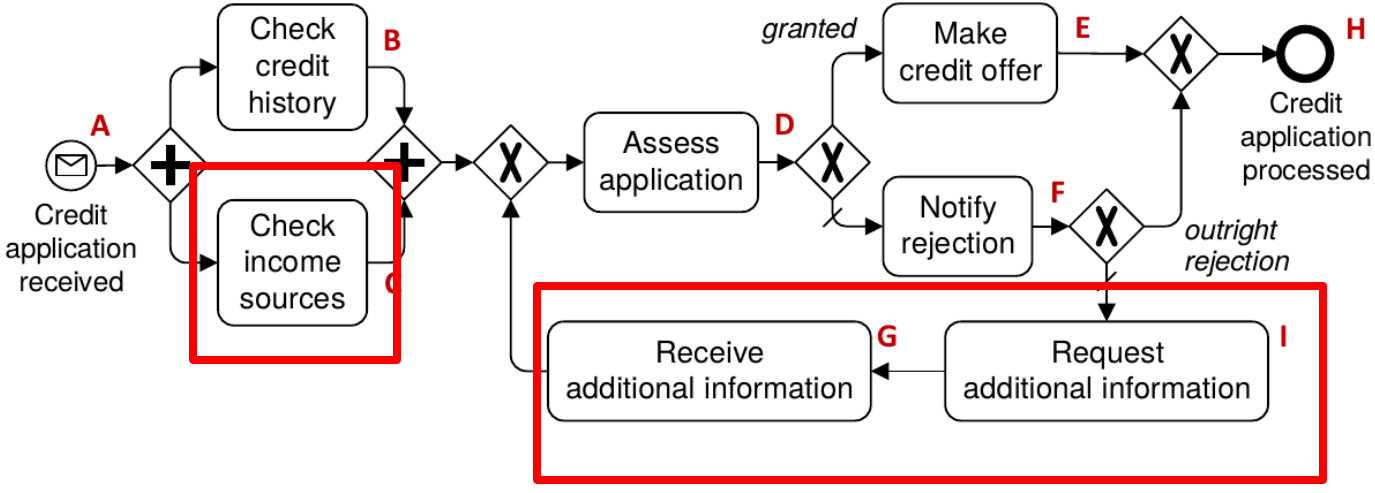
ACBDFEH

ABDEH

ABDFH

¿Qué discrepancias encuentras entre el registro de eventos y el modelo de proceso predefinido?

Para practicar



Event log:

ABCDEH

ACBDEH

ABCDFH

ACBDFH

ACBDFEH

ABDEH

ABDFH

- En un caso del registro de eventos, la tarea E aparece cuando no debe aparecer.
- La tarea C es opcional (es decir, puede omitirse) en el registro, mientras que es obligatoria en el modelo.
- El ciclo que incluye I-G-D-F no se observa en el registro de eventos.

Conformidad de datos y recursos

Las tres perspectivas podrían combinarse.

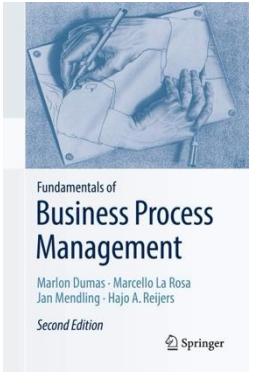
Restricciones de datos y recursos

Por ejemplo:

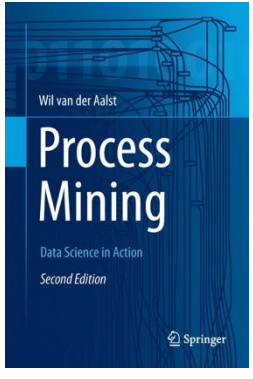
- Reglas específicas al exceder un valor
- Participaciones adicionales en casos críticos (Aprobaciones, autorizaciones)
- Restringir acciones a miembros restringidos.

Normalmente, los permisos se agrupan para roles específicos.

Bibliografía



- Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer Berlin, Heidelberg, 2018.
  - Acceso e-book desde: https://link.springer.com/book/10.1007/978-3-662-56509-4



- Wil M. P Van der Aalst. *Process Mining: Data Science in Action*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.
  - Acceso e-book desde VPN UGR: http://link.springer.com/book/10.1007%2F978-3-662-49851-4

Material muy interesante

- http://www.processmining.org/. Libros, artículos, presentaciones, herramientas para Process Mining.
- http://fundamentals-of-bpm.org/supplementary-material/ presentaciones, modelos de procesos.
- http://www.workflowpatterns.com/. Características comunes que debe reunir cualquier modelo de proceso.

Dr. Adela del Río Ortega
Universidad de Sevilla
adeladelrio@us.es