Capítulo 4

Dependencias Funcionales – Parte 2

Cubrimiento Canónico

- **Problema 1:** dada un conjunto de DF *F* para un problema del mundo real, encontrar la cantidad mínima de DF dentro de *F* sin que se pierda expresividad.
  - Conjuntos de DF pueden tener dependencias redundantes que pueden ser inferidas a partir de otras.
  - **Ejemplo**: $A \rightarrow C$ es redundante en:

$$\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow C\}$$

Cubrimiento Canónico

- Pero se puede hacer algo mejor que lo que pide el problema 1.
  - Una DF puede tener partes de más o atributos de más (llamados **atributos raros**), lo que implica costo adicional en su chequeo.
  - Partes de DF pueden ser redundantes:
    - **Ejemplo**: $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow CD\}$ puede simplificarse a: $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow D\}$.
    - **Ejemplo**: $\{A \rightarrow B, \quad B \rightarrow C, \quad AC \rightarrow D\}$ puede simplificarse a: $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow D\}$.

Cubrimiento Canónico

- Intuitivamente, un **cubrimiento canónico** de F es un conjunto "minimal" de DF equivalente a F, que no tiene DF redundantes o partes redundantes de DFs.
- Sean F y G dos conjuntos de DF. Decimos que F y G son **equivalentes** ($F \equiv G$) if and only if $F \models G$ y $G \models F$.

Cubrimiento Canónico

- Sea *F* un conjunto de DF y la DF $\alpha \rightarrow \beta$ en *F*.
  - El atributo *A* es **raro** in α si $A \in \alpha$
    y $F \equiv (F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.
  - El Atributo *A* is **raro** in β if $A \in \beta$
    y $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\} \equiv F$.
- Las dos definiciones anteriores pueden ser simplificadas

Cubrimiento Canónico

- Sea *F* conjunto de DF y la DF $\alpha \rightarrow \beta$ en *F*.
  - El atributo A is raro en α si $A \in \alpha$
    y $F \models (F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.
  - El atributo *A* es raro en β si $A \in \beta$
    y $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\} \models F$.
- La consecuencia lógica en la dirección opuesta es trivial en cada uno de los casos arriba, debido a que una DF “más fuerte” siempre implica una más débil.

Cubrimiento Canónico

- **Ejemplo**: Dado $F = \{A \rightarrow C, AB \rightarrow C\}$
  - ***¿Hay algún atributo raro en F?***

Cubrimiento Canónico

- **Ejemplo**: Dado $F = \{A \rightarrow C, AB \rightarrow C\}$
  - ***¿Hay algún atributo raro en F?***
  - B is raro in $AB \rightarrow C$ porque $\{A \rightarrow C, AB \rightarrow C\} \models A \rightarrow C$ (I.e. el resultado de tirar B de $AB \rightarrow C$).
- **Ejemplo**: Dado $F = \{A \rightarrow C, AB \rightarrow CD\}$
  - ***¿Hay algún atributo raro en F?***

Cubrimiento Canónico

- **Ejemplo**: Dado $F = \{A \rightarrow C, AB \rightarrow C\}$
  - ***¿Hay algún atributo raro en F?***
  - B is raro in $AB \rightarrow C$ porque $\{A \rightarrow C, AB \rightarrow C\} \models A \rightarrow C$ (I.e. el resultado de tirar B de $AB \rightarrow C$).
- **Ejemplo**: Dado $F = \{A \rightarrow C, AB \rightarrow CD\}$
  - ***¿Hay algún atributo raro en F?***
  - C es raro en $AB \rightarrow CD$ debido a que $AB \rightarrow C$ puede ser inferida incluso después de borrar C.

Cubrimiento Canónico

- Considere *F* conjunto de DFs y la DF $\alpha \rightarrow \beta$ in *F*.
- Para probar si un atributo $A \in \alpha$ es raro en α
  1. Computar $(\alpha - \{A\})^+$ usando las DF de *F*
  2. Chequear que $\beta \subseteq (\alpha - \{A\})^+$; si es cierto entonces *A* es raro
- Veamos que esta receta funciona:

$$
\begin{array}{lll}
 & \beta \subseteq (\alpha - \{A\})^+ & \\
\Rightarrow & F \vdash (\alpha - \{A\}) \rightarrow \beta & \text{por proposición 1} \\
\Rightarrow & F \vdash F - \{\alpha \rightarrow \beta\} \cup \{\alpha - \{A\} \rightarrow \beta\} & \text{def. de } \vdash
\end{array}
$$

Cubrimiento Canónico

- Para probar si un atrtibuto $A \in \beta$ es raro en β
  1. Computar α^+ usando solo las DF en
     $G = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$,
  2. Chequear que $A \in \alpha^+$; si es cierto, entonces A es raro.
- Veamos que esta receta funciona:

$$
\begin{array}{lll}
 & A \in \alpha^+ \text{ bajo } G & \\
\Rightarrow & G \vdash \alpha \rightarrow A & \text{def. de cierre} \\
\Rightarrow & G \vdash \alpha \rightarrow A \wedge & \\
 & G \vdash \alpha \rightarrow \beta - \{A\} & \text{def. de } G \\
\Rightarrow & G \vdash \alpha \rightarrow \beta & \text{por unión} \\
\Rightarrow & G \vdash F & \text{def. de } G \text{ y de } \vdash
\end{array}
$$

Cubrimiento Canónico

- **Ejercicio**: Probar las siguientes afirmaciones usando los métodos de la filmina anterior.
  - $\{A \rightarrow C,\ A\ B \rightarrow C\}$ *B* raro
  - $\{A \rightarrow C,\ A\ B \rightarrow C\ D\}$ *C* raro

Cubrimiento Canónico

- Un **cubrimiento canónico** para F es un conjunto de DF F_c tal que:
  - $F \models F_c$,
  - $F_c \models F$,
  - ninguna DF en F_c contiene un atributo raro, y
  - cada lado izquierdo de una DF en F_c es único.

Cubrimiento Canónico

- Algoritmo para computar el cubrimiento canónico de *F*:
- *Res* := *F*

  repeat

    Use the union rule to replace any dependencies in *Res*

      $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1 \beta_2$

    Find a DF $\alpha \rightarrow \beta$ in *Res* with an

      extraneous attribute either in α or in β

    If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$

  until *Res* does not change

- La regla de unión puede ser aplicable luego de que algunos atributos raros hayan sido borrados, así que tiene que ser reaplicada.
- Este algoritmo tiene el invariante: $Res \models F \land F \models Res$.

Cubrimiento Canónico

- **Ejercicio**: Sea el esquema $R = (A, B, C)$ con DFs:

  $\{A \rightarrow BC,\ B \rightarrow C,\ A \rightarrow B,\ AB \rightarrow C\}$

  – Encontrar recubrimiento canónico usando el algoritmo anterior.

Descomposiciones de Reunión sin Pérdida

- Sea R un esquema de relación. Un conjunto de esquemas de relación $\{R_1, \ldots, R_n\}$ es una **descomposición** de R si

$$R = R_1 \cup \ldots \cup R_n.$$

- **Objetivos** a lograr con una buena descomposición:
  1. Evitar datos redundantes
     - Algoritmos de normalización.
  2. Asegurar que las relaciones entre atributos están representadas.
     - Noción de descomposición de reunión sin pérdida.
  3. Facilitar el chequeo de actualizaciones por violaciones de restricciones de integridad.
     - Noción de preservación de dependencias

Descomposiciones de Reunión sin Pérdida

- Sea R_U esquema universal. Tengo un cierto significado asociado a que una tupla pertenezca a una relación de R_U.
- Si descompongo mal a R_U puedo perder parte de ese significado.

Descomposiciones de Reunión sin Pérdida

- **Ejemplo**: Sea el esquema universal.

  Préstamo = (numSucursal, ciudad, activo, numCliente, numPréstamo, importe)

  - Lo descomponemos en:

    SucursalCliente = (numSucursal, ciudad, activo, numCliente)

    ClientePréstamo = (numCliente, numPréstamo, importe)

  - **Evaluación de la descomposición**: Si tenemos un cliente con varios préstamos en distintas sucursales: no se puede decir el préstamo que pertenece a cada sucursal en la descomposición que tenemos.
    - Luego en la descomposición que tenemos se perdió información con relación al esquema universal.

Descomposiciones de Reunión sin Pérdida

- **Problema:** ¿Cómo formalizar que una descomposición no pierde información?
- **Solución 1**: Seguir el siguiente procedimiento:
  1. Escribir qué significa que una tupla pertenezca a una tabla del esquema universal, expresando todas las relaciones entre atributos relevantes.
  2. Chequear que en la descomposición se mantienen todas esas relaciones entre atributos.
- **Evaluación**: es fácil no darse cuenta de algunas relaciones entre atributos y justo la descomposición que se elige no las tiene en cuenta.
  - El ejemplo anterior justo muestra esto porque se trata de una situación muy particular (que se le puede pasar a cualquiera) que alguien esté en varias sucursales y tenga préstamos en más de una.

Descomposiciones de Reunión sin Pérdida

- **Solución 2**: $r(R)$, R_1 y R_2 descomposición de R, r legal (i.e. cumple restricciones de integridad). Si tiro r y en lugar de r uso $r_i = \Pi_{Ri}(r)$ ($i \in \{1,2\}$); debería poder **reconstruir** r a partir de los r_i
  - ***¿Qué significa reconstruir?***

Descomposiciones de Reunión sin Pérdida

- **Solución 2**: $r(R)$, R_1 y R_2 descomposición de R, r legal (i.e. cumple restricciones de integridad). Si tiro r y en lugar de r uso $r_i = \Pi_{Ri}(r)$ ($i \in \{1,2\}$); debería poder **reconstruir** r a partir de los r_i
  - ***¿Qué significa reconstruir?***
  - En el ejemplo anterior el único atributo en común entre *SucursalCliente* y *ClientePréstamo* es *numCliente*. Luego a lo sumo puedo calcular el natural join de las dos.
  - Reconstruir r significa: $r = \Pi_R\, r_1 \bowtie r_2$
    - Se puede omitir Π_R si el esquema de $r_1 \bowtie r_2$ es R.

Descomposiciones de Reunión sin Pérdida

- Asumimos que *r*(*Préstamo*) viene dada por la siguiente tabla:

r	numSucursal	ciudad	activo	numCliente	numPréstamo	importe
	Centro	Arganzuela	9000000	Santos	P17	1000
	Becenil	Aluche	400000	Santos	P93	500

- Entonces

r_1	numSucursal	ciudad	activo	numCliente
	centro	Arganzuela	9000000	Santos
	Becenil	Aluche	400000	Santos

- Y

r_2	numCliente	numPréstamo	importe
	Santos	P17	1000
	Santos	P93	500

Descomposiciones de Reunión sin Pérdida

- Por otro lado :

$r_1 \bowtie r_2$	numSucursal	ciudad	activo	numCliente	numPréstamo	importe
	Centro	Arganzuela	9000000	Santos	P17	1000
	Centro	Arganzuela	9000000	Santos	P93	500
	Becenil	Aluche	400000	Santos	P93	500
	Becenil	Aluche	400000	Santos	P17	1000

- El ejemplo anterior muestra que al hacer $r_1 \bowtie r_2$ se pueden obtener más tuplas que en r.
  - En ese caso no puedo reconstruir r desde r_1 y r_2.
- **Observación**: Sea $r(R)$ una relación y sea $r_i = \Pi_{Ri}(r)$ $(1 \leq i \leq n)$. $\{r_1, \ldots, r_n\}$ es la BD que resulta de descomponer R en $\{R_1,\ldots,R_n\}$. En general vale $r \subseteq \Pi_R\, r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$

Descomposiciones de Reunión sin Pérdida

- **Definición**: Sea *C* un conjunto de restricciones de integridad de la BD y *R* un esquema de relación. Una descomposición $\{R_1,...,R_n\}$ de *R* es una **descomposición de reunión sin pérdida** si para todas las relaciones *r* del esquema *R* que son legales bajo *C* se cumple que

$$r = \Pi_R\, r_1 \bowtie r_2 \bowtie ... \bowtie r_n$$

  - Recordar que $r_i = \Pi_{Ri}(r)$ para todo i.

Descomposiciones de Reunión sin Pérdida

- ***¿Puede ser necesario hacer la reconstrucción del r?***
  - Podría haber restricciones de integridad que requieran reconstruir el *r* para poder chequearlas.
  - **Ejemplo**: aserción que diga que todo cliente no tenga más de un préstamo por sucursal.
    - *r* lo cumple, pero $r_1 \bowtie r_2$ no.
  - A veces necesito poder reconstruir *r* porque hay consultas que necesitan de *r*.

Descomposiciones de Reunión sin Pérdida

- **Ejemplo**: En el ejemplo de las tablas anterior consideramos la consulta: “Hallar todas las sucursales con préstamos menores a 1000 euros”.
  - Aplicar esta consulta a *r* da diferente que aplicar la misma a $r_1 \bowtie r_2$.
  - En el primer caso da *Becenil* y en el segundo da *Centro* y *Becenil*.

Descomposiciones de Reunión sin Pérdida

- **Proposición 2:** Una descomposición de R en R_1 y R_2 es de reunión sin pérdida si y solo si al menos una de las siguientes DF está en F^+:
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

Descomposiciones de Reunión sin Pérdida

- La prueba se hace para el caso donde $R_1 \cap R_2 \rightarrow R_1 \in F^+$. La prueba del otro caso es similar y por eso se omite.
- Sea *r* legal. Probaremos que $r \supseteq \Pi_R (\Pi_{R1}(r) \bowtie \Pi_{R2}(r))$.

$t \in \Pi_R (\Pi_{R1}(r) \bowtie \Pi_{R2}(r))$

$(\Rightarrow)\ t_1[R_1] = t[R_1] \wedge t_2[R_2] = t[R_2]$ para algún $t_1, t_2 \in r$

$(\Rightarrow)\ t_2[R_2] = t[R_2] \wedge t_1[R_1] = t[R_1] \wedge$
$t_2[R_1 \cap R_2] = t_1[R_1 \cap R_2] \wedge t_1, t_2 \in r$

$(\Rightarrow)\ t_2[R_2] = t[R_2] \wedge t_2[R_1] = t_1[R_1] \wedge$ $F \models R_1 \cap R_2 \rightarrow R_1$
$t_1[R_1] = t[R_1] \wedge t_2 \in r$ y r legal bajo F

$(\Rightarrow)\ t_2[R_2] = t[R_2] \wedge t_2[R_1] = t[R_1] \wedge t_2 \in r$

$(\Rightarrow)\ t_2[R] = t[R] \wedge t_2 \in r$ $R = R_1 \cup R_2$

$(\Rightarrow)\ t = t_2 \wedge t_2 \in r$

$(\Rightarrow)\ t \in r$

Descomposiciones de Reunión sin Pérdida

- **Ejercicio**: Si considero la descomposición de *Préstamo* en:
  - *Sucursal* = (*numSucursal*, *ciudad*, *activo*)
  - *InfoPréstamo* = (*numSucursal*, *numCliente*, *numPréstamo*, *importe*)
  - **¿Se puede recuperar *r(Préstamo)* legal a partir de r_1*(Sucursal)* y r_2*(InfoPréstamo)*?**

Descomposiciones de Reunión sin Pérdida

- **Ejercicio**: Si considero la descomposición de *Préstamo* en:
  - *Sucursal* = (*numSucursal*, *ciudad*, *activo*)
  - *InfoPréstamo* = (*numSucursal*, *numCliente*, *numPréstamo*, *importe*)
  - **¿Se puede recuperar *r(Préstamo)* legal a partir de r_1(*Sucursal*) y r_2(*InfoPréstamo*)?**
  - Notar que para *Sucursal*, se tiene la DF:

    numSucursal → *ciudad*, *activo*
  - Obviamente que *numSucursal* → *ciudad*, *activo, numSucursal*

Descomposiciones de Reunión sin Pérdida

- **Ejercicio**: Sea $R = (A, B, C)$ esquema con DFs:

  $F = (A \rightarrow B, B \rightarrow C)$.

  – Sea $R_1 = (A, B)$, $R_2 = (B, C)$ descomposición de R.

  – Probar que esa es una descomposición de reunión sin pérdida.

Descomposiciones de Reunión sin Pérdida

- **Ejercicio**: ¿Sea R= (A, B), R_1 = (A), R_2 = (B), será esa una descomposición de reunión sin pérdida?
- **Ejercicio**: Sean R = (A, B, C, D), F = {$A \rightarrow BC$, $D \rightarrow A$, $B \rightarrow D$} y Q = {(A, B), (A, C), (B, D)} una descomposición de R. ¿es Q de reunión sin pérdida? Justifique su respuesta.

Preservación de Dependencias

- Uno de los objetivos a lograr con una buena descomposición del esquema universal es el **chequeo económico de las DF**.
- Chequear algunas DF obligan a construir reuniones naturales de tablas y luego chequearlas.
  - Esto suele ser muy costoso.
  - A uno le gustaría que no haya que calcular reuniones naturales para chequear DF.

Preservación de Dependencias

- **Problema**: ¿Cuáles son las DF que se pueden chequear económicamente?
- **Proposición 3**: $R_1,\ldots, R_n$, descomposición de R, F conjunto de DFs, $r(R)$ legal, $\alpha, \beta \subseteq R_i$, entonces: $\alpha \rightarrow \beta$ se cumple en $\prod_{Ri}(r)$ si y solo si $\alpha \rightarrow \beta$ se cumple en r .
- **Respuesta**: dependencias con atributos en miembros de la descomposición se pueden chequear económicamente.
  - No hace falta calcular reuniones naturales para su chequeo.
  - Se las caracteriza en la definición abajo.

Preservación de Dependencias

- **¿Cómo formalizar las DF que se pueden chequear económicamente?**
- **Definición**: Sea F un conjunto de DF del esquema R y R_1, R_2,.., R_n , una descomposición de R. La **restricción** de F a R_i se denota F_i y es el conjunto de todas las DF de F^+ que incluyen solo atributos de R_i. Formalmente:

$$F_i = \{\alpha \rightarrow \beta \in F^+ : \alpha, \beta \subseteq R_i\} .$$

- **Meta**: se debería tener un conjunto de DFs representativas del problema del mundo real que se chequeen en las tablas de la BD.
- **¿Cómo formalizar esta meta?**

Preservación de Dependencias

- **Solución**: Sea $F' = F_1 \cup F_2 \cup \ldots \cup F_n$. Se dice que las descomposiciones donde se cumple $F'^{+} = F^{+}$ son **descomposiciones que conservan las dependencias**.
  - Si para todo *i*, F_i se cumple en $\prod_{Ri}(r)$, entonces F se cumple en r.
- **Nota**: para cada F_i se puede calcular un recubrimiento canónico y la unión de todos esos recubrimientos canónicos será el conjunto de dependencias a usar que son chequeables eficientemente.

Preservación de Dependencias

- **Observación**: Si se puede comprobar cada miembro de F en alguna de las relaciones de la descomposición, entonces la descomposición trivialmente preserva las dependencias.
- **Ejercicio**: Sea $R = (A, B, C)$ con conjunto de DFs $F = \{A \rightarrow B, B \rightarrow C\}$, y sea la descomposición de R:

$$R_1 = (A, B), R_2 = (B, C) .$$

¿Esta descomposición preserva las dependencias?

Preservación de Dependencias

- Hay casos en los que a pesar que una descomposición preserva las dependencias hay un miembro de *F* que no puede verificarse en ninguna de las tablas de la BD.

Preservación de Dependencias

- Para la comprobación de la conservación de las dependencias se usa el siguiente método: Se aplica el siguiente procedimiento a cada DF $\alpha \rightarrow \beta \in F$:
  - *result* = α
    while (changes to *result*) do
      for each R_i in the decomposition
        $t = (result \cap R_i)^+ \cap R_i$
        $result = result \cup t$
  - Si *result* contiene todos los atributos en β, entonces la DF $\alpha \rightarrow \beta$ es preservada.

Preservación de Dependencias

- Aplicamos la prueba a todas las DF de *F* para chequear si una decomposición preserva las dependencias.
- Este procedimiento toma tiempo polinomial, en lugar del tiempo exponencial requerido para computar F^+ y $(F_1 \cup F_2 \cup \ldots \cup F_n)^+$

Preservación de Dependencias

- Sea $\alpha \rightarrow \beta \in F$. $F' \vdash \alpha \rightarrow \beta$ si y solo si $\beta \subseteq \alpha_{F'}^{+}$.
  - Si probamos que el algoritmo anterior computa $\alpha_{F'}^{+}$, entonces la receta anterior es correcta.
- Probaremos que el algoritmo anterior se obtiene luego de aplicar transformaciones correctas al algoritmo que calcula $\alpha_{F'}^{+}$.

Preservación de Dependencias

El algoritmo que calcula α^+ respecto de F' es:

$res := \alpha$;
while cambios en res **do**
    for each $\beta \rightarrow \gamma$ en F' **do**
        if $\beta \subseteq res$ **then** $res := res \cup \gamma$
    od
od

Usando que $F' = F_1 \cup ... \cup F_n$ el algoritmo anterior es equivalente al siguiente:

Preservación de Dependencias

```
res := α;
while cambios en res do
    for each i = 1 to n do
        for each β → γ en Fi do
            if β ⊆ res then res := res ∪ γ
        od
    od
od
```

Observamos que en el algoritmo anterior el for de más adentro calcula

$$\cup\{\gamma \mid \beta \rightarrow \gamma \in F_i \wedge \beta \subseteq res\} \ .$$

Luego el algoritmo anterior es equivalente a:

Preservación de Dependencias

$res := \alpha$;
while cambios en res **do**
    for each $i := 1$ **to** n **do**
        $t := \cup\{\gamma \mid \beta \rightarrow \gamma \in F_i \wedge \beta \subseteq res\}$
        $res := res \cup t$
        od
    od
od

Si probamos que

$$\cup\{\gamma \mid \beta \rightarrow \gamma \in F_i \wedge \beta \subseteq res\} = (res \cap R_i)^+ \cap R_i \ .$$

entonces estamos listos.

Preservación de Dependencias

Ahora probamos la primera inclusión ($\subseteq$).

$$\begin{aligned}
& \beta \to \gamma \in F_i \wedge \beta \subseteq res \\
\Rightarrow \quad & \beta = \beta \cap R_i \subseteq res \cap R_i \wedge \beta \to \gamma \in F_i \\
\Rightarrow \quad & \beta^+ \subseteq (res \cap R_i)^+ \wedge F \vdash \beta \to \gamma && + \text{ es monótona} \\
\Rightarrow \quad & \gamma \subseteq \beta^+ \subseteq (res \cap R_i)^+ && \text{proposición 1}
\end{aligned}$$

Ahora probamos la otra inclusión.

$$\begin{aligned}
& A \in (res \cap R_i)^+ \cap R_i \\
\Rightarrow \quad & A \in (res \cap R_i)^+ \wedge A \in R_i \\
\Rightarrow \quad & F \vdash (res \cap R_i) \to A \wedge A \in R_i && \text{proposición 1} \\
\Rightarrow \quad & (res \cap R_i) \to A \in F^+ \wedge A \in R_i && \text{def. de cierre} \\
\Rightarrow \quad & (res \cap R_i) \to A \in F_i \wedge res \cap R_i \subseteq res && \text{def. de } F_i \\
\Rightarrow \quad & A \in \cup\{\gamma \mid \beta \to \gamma \in F_i \wedge \beta \subseteq res\} && \text{teoría de conjuntos}
\end{aligned}$$

Preservación de Dependencias

- **Ejercicio**: Sea $R = (A, B, C)$ con conjunto de DFs $F = \{A \rightarrow B, B \rightarrow C\}$, y sea la descomposición de R:

$$R_1 = (A, B), R_2 = (A, C) .$$

  - ¿Será que esa descomposición conserva las dependencias?
    - Resolver primero usando definición de preservación de dependencias.
    - Resolver luego usando el algoritmo anterior.