Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Práctico de IBC

Federico De Rocco

Director: Vanina Martinez

Buenos Aires, 2020

Índice general

1. PRÁCTICO TEMA 1

1. Suponiendo la premisa como cierta, se puede asumir que programas triviales (como por ejemplo el "Hola Mundo") no presentan un comportamiento que podríamos llamar inteligente, ya que este es altamente determinístico. Pero este comportamiento no siempre se puede analizar de esta forma. El programa podría ser muy complejo, de tal manera que su determinismo no estuviera claro, ni siquiera para los programadores. Pero todo esto es suponiendo que este sea determinístico, uno podría escribir una pieza de software que tuviera comportamiento no determinístico. Un ejemplo de esto sería que el programa seleccionara entre varios comportamientos distintos en base a valores generados aleatoriamente. Deberíamos preguntarnos también, si el hecho de tener comportamiento que este influenciado cualquier forma implica no tener inteligencia, lo cual es lo que asume esta frase. A mi entender esto no es cierto, y considero que los seres humanos poseamos algunas formas de este.

2. Existen experimentos como la caja de Skinner, que han demostrado que, sometiendo a estímulos apropiados se puede lograr que un animal desarrolle un comportamiento determinado. Pero esto no significa que Skinner u otros científicos tengan la capacidad de predecir todos los elementos de como los animales actúan. Usar a los "genes", como una posible forma de insistir en que los animales no son inteligentes es ridículo porque, de primeras los seres humanos seriamos en el mejor caso, versiones más complejas de ellos y considero que decir que los seres humanos no podemos ser inteligentes sería dejar de lado la metáfora. Pero en segundo lugar, al igual que las computadoras, lo innato en los animales podría sugerir comportamiento que fuera muy difícil o casi imposible de determinar.

3. No solo el sistema puede reconocer y entender información tal como el tráfico aéreo, sino que puede reaccionar a las situaciones que esta información indica. Su comportamiento queda determinado por decisiones tomadas por él mismo usando la información que recibe. Esto muestra un programa capaz de adaptarse y tomar decisiones con criterios determinados. Si se está dispuesto a clasificar este tipo de comportamiento como inteligente, entonces estamos ante un sistema que lo es.

En la naturaleza, las aves voladoras poseen características similares, ellas no siguen una ruta fija para moverse, cazar, migrar, etc. Divergen en ellas por decisiones tomadas de acuerdo con los elementos en el entorno que pueden percibir, el viento y el clima por ejemplo. Pero estas también pueden enfrentarse a situaciones de las cuales no tengan una respuesta optima o no sepan percibirla, por ejemplo si un avión vuela contra ellas o si un cazador les dispara en el aire.

4. Dado que existe un problema que no puede ser resuelto utilizando software, debo entender que la única razón por la que esto puede impedir el desarrollo de la IA es que esta no podría resolver estos problemas. Si entendemos que la IA debe ser capaz de replicar cualquier comportamiento inteligente que un ser humano o animal puede realizar, entonces efectivamente podemos decir que esta sería imposible porque las piezas de software no podrían resolver estos problemas. Pero esto solo tiene sentido si tenemos este como

la definición de inteligencia artificial, si concedimos que una de estas puede dedicarse a resolver problemas específicos, como la navegación en el avión del ejercicio anterior, entonces no tenemos este problema. Además la mayoría de definiciones de IA implican que el programa debe operar bajo una metáfora del pensamiento, la cual no tiene porque ser biologicamente observable, para considerarse IA. Noten que esto implica la noción de inteligencia, no como su totalidad, sino como posibles formas de funcionamiento que pueda argumentarse que esta pueda operar. Sugiriendo que la idea es que exhiba comportamiento inteligente según esta metáfora, no que sea irreconocible de las capacidades humanas. Incluso podríamos decir que si abstraemos todos los elementos de inteligencia que no son suficientes para resolver estos problemas no computables, y creamos una IA a partir de estos, tendríamos un aproximado a una IA general.

5.

a) Por si mismo es capaz de realizar varias actividades asociadas con el pensamiento; la de lectura y análisis de un mapa, el análisis de información dinámica como el tráfico, el catalogar y estimar tiempos de distintos medios de transporte y el comunicar esta información a pedido. Todo esto con el objetivo de obtener las mejores rutas a pedido. Es una instancia de IA en el sentido que pretende emular una actividad, hasta hace poco puramente humana, tal es como la planificación de ruta en un entorno de transporte humano.

b) Este sistema realiza la actividad de reconocer patrones simbólicos, mediante un escaner, traducirlos a un valor número y a este asociarlo a un producto con su precio y otros datos. Sin duda, la actividad que más resalta es la de reconocimiento simbólico; particularmente la asociación de un símbolo a un valor numérico o, extendiendo la metáfora, a una idea es una de las actitudes más comunes de la inteligencia, solo hay que ver a las letras de el lenguaje escrito que sea para reconocer esto. Incluso los números, que en diferentes lugares o tiempos se representan con símbolos distintos.

c) En este caso tenemos un sistema de reconocimiento de voz. Se podría decir que este es una instancia de IA en el sentido de que la actividad que realiza está asociada al lenguaje verbal, el cual es parte de la forma en la que los humanos piensan. Aunque a su vez se podría decir que también los animales reconocen la voz e incluso pueden seguir ordenes comunicadas con esta, aunque no todos los animales pueden. Hasta antes de un sistema de reconocimiento de voz, la actividad solo podía realizarse a travez de seres vivientes específicos, dadas sus capacidades para escuchar y reconocer diferencias en el sonido.

d) Siendo el lenguaje escrito un fenómeno, posiblemente accidental a diferencia del verbal pero, sin duda propio del pensamiento humano. Esta sería entonces una máquina sin conciencia y carente de las cualidades innatas de los seres humanos que puede reconocer patrones en el habla para asociarlos con instrucciones que normalmente debería introducirse manualmente.

e) Similar al sistema de GPS, la idea es poder encontrar el mejor recorrido, teniendo en cuenta la disponibilidad de los elementos que son parte de los caminos potenciales. Es cierto que el camino no se refiere a lo mismo, ya que son paquetes de datos y no una persona, disponibilidad de nodos y no tráfico o disponibilidad de transportes, en última

instancia es inteligente por razones similares.

2. PRÁCTICO TEMA 2: AGENTES

1.

```
funcion TreeSearch(Raiz, ArbolBusqueda, Estrategia) {
  Frontera = []
  Frontera.add(Raiz)
  SelectElement = Raiz
  PotencialesCaminos = [[Raiz]]

  while (!esMeta(SelectElement)) {

    Frontera = Estrategia.nuevaFrontera(ArbolBusqueda, Frontera, SelectElement)

    if (Frontera.lenght == 0) {
      return failure
    }

    SelectElement = Estrategia.select(ArbolBusqueda, Frontera)

    AgregarAPotencialesCaminos(ArbolBusqueda, PotencialesCaminos, SelectElement)
  }

  AgregarAPotencialesCaminos(ArbolBusqueda, PotencialesCaminos, SelectElement)

  Camino = PotencialesCaminos.last()

  return Camino

}

function AgregarAPotencialesCaminos(ArbolBusqueda, PotencialesCaminos, Elemento) {

  foreach PotencialCamino en PotencialesCaminos {
    UltimoElementoDelCamino = PotencialCamino.lastElement()

    if (Elemento en ArbolBusqueda.vecinos(UltimoElementoDelCamino)) {
      UltimoElementoDelCamino.add(Elemento)
    }
  }


}

```

2.

```
funcion PrimeroProfundidad(Raiz, ArbolBusqueda) {
  Frontera = []
  Frontera.add(Raiz)
  SelectElement = Raiz
  PotencialesCaminos = [[Raiz]]
  Visitados = []


  while (!esMeta(SelectElement)) {

    Vecinos = ArbolBusqueda.vecinos(SelectElement)

    Frontera.remove(SelectElement)

    if (Vecinos != []) {
      Frontera = Vecinos + Frontera
    }

    if (Frontera.lenght == 0) {
      return failure
    }

    SelectElement = Frontera.first()

    AgregarAPotencialesCaminos(ArbolBusqueda, PotencialesCaminos,
SelectElement)
  }

  AgregarAPotencialesCaminos(ArbolBusqueda, PotencialesCaminos,
SelectElement)

  Camino = PotencialesCaminos.last()

  return Camino

}


funcion PrimeroAncho(Raiz, ArbolBusqueda) {
  Frontera = []
  Frontera.add(Raiz)
  SelectElement = Raiz
  PotencialesCaminos = [[Raiz]]

  while (!esMeta(SelectElement)) {

    Vecinos = ArbolBusqueda.vecinos(SelectElement)

    Frontera.remove(SelectElement)

    Frontera = Frontera + Vecinos

    if (Frontera.lenght == 0) {
      return failure
    }

    SelectElement = Frontera.first()

```

```
57          AgregarAPotencialesCaminos(ArbolBusqueda, PotencialesCaminos,
    SelectElement)
58        }
59
60        AgregarAPotencialesCaminos(ArbolBusqueda, PotencialesCaminos,
    SelectElement)
61
62        Camino = PotencialesCaminos.last()
63
64        return Camino
65
66      }
67
68
```

3. El algoritmo de búsqueda Iterative deeping es completo. La razón es que al correr búsqueda en profundidad repetidamente con niveles de profundidades incrementales, se tiene efectivamente un algoritmo del cual el orden conmutativo de los nodos que visita es el de búsqueda en lo ancho. Este último es un algoritmo de búsqueda completo, con lo cual, Iterative deeping también lo és.

4.

Autonomía: La capacidad de elegir si actuar o no, y en que forma hacerlo.

Agente reflejo: Es un tipo de agente que observa y si ve algo determinado realiza una acción. Caso contrario, no hace nada o hace otra cosa. Esto en base a reglas de condición-acción. No posee un estado interno.

Agente basado en metas: Posee un estado interno que le permite determinar las consecuencias de realizar una acción. A su vez posee metas a alcanzar. De todas las acciones posibles, descarta las que no tienen que ver con las metas que posee, sin importar cual de las que la lleva a la meta es la mejor.

Agente basado en utilidades: En lugar de metas se tiene una utilidad, en estos casos puede haber más de una acción que lleva a la meta. La utilidad sirve para que se tomen las "mejores" decisiones.

5.

```
1     funcion AgenteBasadoEnMetas(DatosDeSensores, ComoEvolucionaElMundo,
    ConsecuenciasDeLasAcciones, Estado, Metas) : Accion {
2       Acciones =  ObtenerPosiblesAcciones(DatosDeSensores,
    ComoEvolucionaElMundo, ConsecuenciasDeLasAcciones, Estado)
3
4       foreach Accion en Acciones {
5         ConsecuenciasDeAccion = ObtenerConsecuenciasDeAccion(
    ComoEvolucionaElMundo, ConsecuenciasDeLasAcciones, Estado, Accion)
6         if (EstaMasCercaOEsMeta(ConsecuenciasDeAccion, Metas, Accion)) {
7           return Accion
8         }
```

```
9       }
10
11      return AccionNula //No hace nada
12    }
13
14
15    funcion AgenteBasadoEnUtilidades(DatosDeSensores, ComoEvolucionaElMundo
      , ConsecuenciasDeLasAcciones, Estado, Utilidades) : Accion {
16      Acciones =  ObtenerPosiblesAcciones(DatosDeSensores,
      ComoEvolucionaElMundo, ConsecuenciasDeLasAcciones, Estado)
17      AccionATomar = AccionNula // Tiene Valor 0
18
19      foreach Accion en Acciones {
20        ConsecuenciasDeAccionActual = ObtenerConsecuenciasDeAccion(
      ComoEvolucionaElMundo, ConsecuenciasDeLasAcciones, Estado, AccionATomar
      )
21
22        ConsecuenciasDeAccion = ObtenerConsecuenciasDeAccion(
      ComoEvolucionaElMundo, ConsecuenciasDeLasAcciones, Estado, Accion)
23
24        ValorDeEstadoAlQueLlegoActual = ValorDeEstado(Utilidades,
      ConsecuenciasDeAccionActual)
25
26        ValorDeEstadoAlQueLlego = ValorDeEstado(Utilidades,
      ConsecuenciasDeAccion)
27
28        if (ValorDeEstadoAlQueLlegoActual < ValorDeEstadoAlQueLlego) {
29          AccionATomar = Accion
30        }
31
32      }
33
34      return AccionATomar
35    }
36
37
38
```

6. Es una instancia de agente reactivo con estado interno, solo tiene dos acciones (apagar y encender) y si bien pueden haber muchas temperaturas diferentes; solo le interesan las que están por debajo de tres grados de la temperatura seteada o por encima de esta. Se necesita un estado para la temperatura seteada y para el estado actual de prendido o apagado, ya que da a entender que la acción apagar solamente está para cuando se encuentra encendida y viceversa. No hay metas, ya que solamente va cambiando entre dos estados sin fin. Por lo tanto, tampoco puede ser de utilidad.

3. PRÁCTICO TEMA 3

1. Las creencias del termostato son que siempre sabe cuando la temperatura, además de saber si la caldera está o no apagada. Sus deseo es que la temperatura siempre este entre 3 grados por debajo o por arriba de la temperatura dada. Su intención es que siempre que la temperatura este por debajo, encenderá la caldera para aumentarla, y la apagará cuando este por encima para disminuirla.

2. La cantidad de semillas en una naranja específica.

Se que las frutas son frutas porque tienen semillas, mi naranja tiene semillas; por lo tanto, mi naranja es una fruta.

En un grupo de amigos cada uno conoce los nombres, y o formas de referirse, a los otros.

Conocimiento interno de conocimiento externo: conocimiento como una relación entre un sujeto cognitivo y partes de la realidad. El conocimiento de que un perro desconocido mostrando los dientes es una señal de amenaza.

3. Supongamos dos KBs, una cita a todas las personas registradas que son programadores, la segunda lista a todas las personas registradas que son mujeres.

Persona	Especialidad
Karen Sanchez	Programadora
Karen Carles	Programadora

Persona	Especialidad
Karen Sanchez	Programadora
Karen Carles	Programadora
Emma Perez	Medica

La segunda base claramente contiene a la primera. Si tuviéramos un programador hombre, que no estuviera en la primer base, entonces la monotonía derivada de la suposición de mundo cerrado no se cumpliría porque la segunda no lo puede contener.

4. Mundo de bloques: modele el mundo de bloques en lógica proposicional. Debe modelar el estado de la mesa: tenemos 3 bloques (y solo 3) A, B y C. Debemos poder expresar que los bloques están sobre la mesa, o sobre otro bloque. También queremos expresar la siguiente propiedad (para los tres bloques): un bloque se puede desapilar siempre y cuando esté sobre otro bloque y no tenga un bloque encima. ¿Cual sería la KB que represente como estado inicial el hecho de que A está sobre la mesa, B está sobre A y C está sobre la mesa? Muestre que para la consulta "¿se puede desapilar A?", la respuesta es negativa, usando ambos algoritmos de inferencia (encadenamiento hacia atrás y hacia adelante). Asuma CWA. Recuerde que además de las reglas que defina en la teoría, puede utilizar las reglas de inferencia y equivalencias lógicas para la lógica proposicional (Artificial Intelligence A

Modern Approach. Sección 7 – tercera Edición). Una lista completa de las reglas puede encontrarse en: $https://es.wikipedia.org/wiki/Anexo:Reglas_de_inferencia$

Si X en mesa lo que estoy diciendo es que el bloque representado por X está sobre la mesa. Si X en Y, con $X \neq Y$, digo que el bloque X está sobre el Y. Además de esto tengo la siguientes reglas:

$\neg X \text{ en } X$

$\neg X \text{ en } Y \land \neg X \text{ en } Z \land Y \neq Z \land X \neq Y \land X \neq Z \iff X \text{ en } mesa$

$X = A \lor X = B \lor X = C$

$Y = A \lor Y = B \lor Y = C$

$Z = A \lor Z = B \lor Z = C$

La primera dice que un bloque no puede estar encima de si mismo, y la segunda que si el bloque X no está en encima de Y o Z, siendo estos tres bloques distintos, entonces X está sobre la mesa. Por último los valores de X, Y y Z se restringen a los tres bloques. Debemos considerar también el desapilar. Para eso tenemos esta regla:

$X \textit{ se puede desapilar} \iff \neg X \text{ en } mesa \land \neg Y \text{ en } X \land \neg Z \text{ en } X \land Y \neq Z \land X \neq Y \land X \neq Z$

Veremos ahora la KB sería: $\{B \text{ en } A\}$. Dado que al no decir si A o C están sobre ningún bloque se pueden inferir, asumiendo CWA, que están sobre la mesa. Ahora procederemos a responder la consulta de si se puede desapilar A.

Encadenamiento para adelante:

Por CWA, podemos deducir los siguientes hechos:

1. $\neg A \text{ en } B$
2. $\neg A \text{ en } C$
3. $B \text{ en } A$
4. $\neg B \text{ en } C$
5. $\neg C \text{ en } A$
6. $\neg C \text{ en } B$
7. $A \neq B$
8. $A \neq C$
9. $B \neq C$

Si combinamos 1, 2, 7, 8 y 9 tenemos:

$\neg A \,\text{en}\, B \wedge \neg A \,\text{en}\, C \wedge B \neq C \wedge A \neq B \wedge A \neq C$

Según la regla de mesa, esto implica que $A \,\text{en}\, mesa$. Si a su vez, combinamos está conclusión con los hechos 3, 5, 7, 8, 9, tenemos lo siguiente:

$\neg A \,\text{en}\, mesa \wedge \neg B \,\text{en}\, A \wedge \neg C \,\text{en}\, A \wedge B \neq C \wedge A \neq B \wedge A \neq C$

Y con esto podemos derivar que A se puede desapilar por la regla de desapilación. Pero esto se deduce como falso ya que por un lado $A \,\text{en}\, mesa$, por la regla anterior, y $B \,\text{en}\, A$, este es un hecho.

Encadenamiento para atrás:

Comenzando por la regla del desapilado, que es lo que queremos probar.

$\neg A \,\text{en}\, mesa \wedge \neg B \,\text{en}\, A \wedge \neg C \,\text{en}\, A \wedge B \neq C \wedge A \neq B \wedge A \neq C \iff A \textit{ se puede desapilar}$

Por el lado izquierdo, tenemos $\neg A \,\text{en}\, mesa$. Usando la regla de la mesa podemos ver que:

$\neg A \,\text{en}\, B \wedge \neg A \,\text{en}\, C \wedge B \neq C \wedge A \neq B \wedge A \neq C \iff A \,\text{en}\, mesa$

Se puede ver que a la izquierda cada uno de los literales es un hecho de la lista; 1, 2, 7, 8 y 9. Por lo tanto podemos concluir que $A \,\text{en}\, mesa$. Pero esto no es lo que queríamos, entonces tenemos que A no se puede dasapilar.

5. ¿Qué tipo de razonamiento cree usted que lleva a cabo un algoritmo de clasificación basado en datos, de acuerdo a los tipos vistos en clase?

6.

- Las playas están compuestas de arena.
- Algunas playas en el mundo están compuestas por una mezcla de arena y otros elementos, como botellas de plástico. Entre ellas se encuentra la playa de Mar del Plata.
- La playa donde fui de vacaciones está compuesta por pura arena.

Como se puede ver este es un ejemplo de razonamiento no-monótono ya que se está llegando a una conclusión a partir de información incompleta. Si se agrega la frase "La playa donde fui de vacaciones fue la Mar del Plata", se puede ver la conclusión debe retractarse y se debería cambiar por "La playa donde fui de vacaciones no está compuesta por pura arena".

Ahora si queremos representar esto con lógica clásica:

"Las playas están compuestas de arena"

$\forall X(esPlaya(X) \wedge \neg excepcion(X) \longrightarrow elContenidoEsArena(X))$

Tenemos que agregar una regla para las excepciones:

$\forall X(excepcion(X) \iff \neg marDelPlata(X) \wedge ...)$

No sé todas las excepciones y para poder llegar a la conclusión de que el contenido es arena, debería afirmar que la playa no es ninguna de las excepciones. Con esto concluyo que no puedo realizar una representación de este razonamiento con lógica clásica.

Dado que la información está incompleta el razonamiento presentado es no-monótono.

4. PRÁCTICO TEMA 4

1. AbueloOAbuela $\equiv$ Persona $\sqcap \exists$hijo-de.PadreOMadre

AbuelaMaterna $\equiv$ Mujer $\sqcap$ AbueloOAbuela

PadreOMadreDeMasDeDos $\equiv$ Persona $\sqcap >=2$ (hijo-de.Persona)

HermanoOHermana $\equiv \exists$hijo-de^{-}1.PadreOMadreDeMasDeDos

SinHijos $\equiv$ Persona $\sqcap \neg\exists$hijo-de.Persona

TioTia $\equiv$ Persona $\sqcap$ HermanoOHermana $\sqcap\exists$hijo-de^{-}1.AbueloOAbuela

TioSinHijos $\equiv$ TioTia $\sqcap$ SinHijos

AbueloOAbuelaDeMasDeDos $\equiv$ PadreOMadreDeMasDeDos $\sqcap$ AbueloOAbuela

PadreDelSobrinoDeUnaNuera $\equiv \exists$ hijo-de^{-}1.AbueloOAbuelaDeMasDeDos

SobrinoDeUnaNuera $\equiv$ Hombre $\sqcap \exists$hijo-de^{-}1.(PadreDelSobrinoDeUnaNuera)

PadreConAlMenos3HijosDosMujeres $\equiv$ Persona $\sqcap >=3$ (hijo-de.Persona) $\sqcap >=2$ (hijo-de.Mujer)

2. Teniendo el siguiente programa en Datalog:

1. $father(alice, bob)$.
2. $mother(alice, carla)$.
3. $mother(evan, carla)$.
4. $father(carla, david)$.
5. $parent(X, Y) \leftarrow father(X, Y)$
6. $parent(X, Y) \leftarrow mother(X, Y)$
7. $ancestor(X, Y) \leftarrow parent(X, Y)$
8. $ancestor(X, Z) \leftarrow parent(X, Y) \wedge ancestor(Y, Z)$

Para derivar $parent(evan, david)$ uno debe apoyarse en las reglas 5 y 6 sería necesario que $father(evan, david)$. o $father(evan, david)$. existieran. Como no es el caso, $parent(evan, david)$ no se puede derivar.

Para el caso de $ancestro(carla, david)$, si es posible la derivación. A continuación, se presenta el árbol:

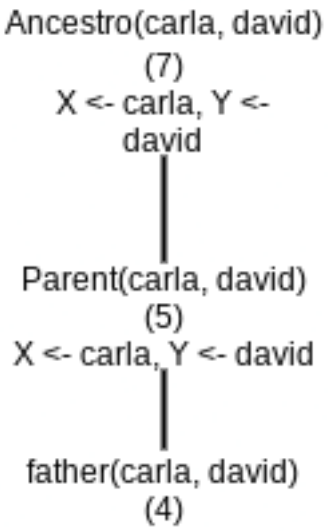

Fig. 4.1: Árbol de derivación

3. Dada la consulta conjunctiva: Q(X,Y) = ∃Z.parent (X,Y) ⋀ parent (Z,Y) ¿Qué regla debemos agregarle a P (slide 49)? ¿Cual sería el conjunto de respuesta? Verifiquelo en ABCDatalog.

Debemos agregar la regla:

$Q(X, Y) \leftarrow parent(X, Y) \wedge parent(Z, Y).$

Este es el resultado de la consulta Q(X, Y) sería el equivalente a $parent(X, Y)$; en ABC-Datalog:

$\{Q(alice, bob), Q(alice, carla), Q(carla, david), Q(evan, carla)\}$

4. $Q_1 \leftarrow parent(X, Y) \wedge parent(Z, Y).$

5. De esta forma queda el mundo de bloques modelado en Datalog:

$estaSobre(b, a).$

$tresBloques(a).\ tresBloques(b).\ tresBloques(c).$

$estaSobreTresBloques(X, Y) \leftarrow tresBloques(X), tresBloques(Y), X \neq Y, estaSobre(X, Y).$

$estaSobreMesa(X) \leftarrow tresBloques(X), tresBloques(Y), tresBloques(Z), X \neq Y, X \neq Z, Y \neq Z, notestaSobre(X, Y), notestaSobre(X, Z).$

$estaApiladoSobre(X, Y) \leftarrow estaApiladoSobreInmediatamente(X, Y).estaApiladoSobre(X, Y) \leftarrow estaApiladoSobre(X, Z), estaApiladoSobre(Z, Y).$

$estaDebajo(X, Y) \leftarrow estaApiladoSobre(Y, X).$

$estaApiladoSobreInmediatamente(X, Y) \leftarrow estaSobreTresBloques(X, Y).$

$estaDebajoInmediatamente(X, Y) \leftarrow estaSobreTresBloques(Y, X).$

$puedeDesapilar(X) \leftarrow tresBloques(X), tresBloques(Y), tresBloques(Z), X \neq Y, X \neq Z, Y \neq Z, notestaSobreMesa(X), notestaSobre(Y, X), notestaSobre(Z, X).$

6. Proponga un ejemplo de dos esquemas de bases de datos (fuente y destino), muestre las s-t-TGDs necesarias para pasar de uno a otro, similar a slide 91, explique la transformación necesaria. Defina una instancia de bases de datos en el esquema fuente y muestre cual seria la solución (la instancia destino) para el problema de intercambio de datos.

Esquema fuente: relaciones transacción, usuario, transacción_cuenta, transacción_usuario.
Esquema destino: relaciones préstamo, usuario.

El esquema fuente la transacción está compuesta por un id, un id de usuario, un importe y una fecha. También posee un valor que le permite distinguir entre los distintos tipos de transacciones, entre ellas los préstamos. La relación transacción_cuenta, agrupa a una transacción con la cuenta que fue utilizada en su proceso, transacción_usuario hace lo mismo con los usuarios. El préstamo por su lado tiene los mismos datos que la transacción, descartando el que indica su tipo y además posee un contrato asociado que deberá generarse y que no forma parte del esquema origen. Los usuarios son equivalentes.

$\forall\, TID, UID, IMPORTE, FECHA, CUENTA$
$(transaccion(TDI, "ES\_PRESTAMO", UID, IMPORTE, FECHA) \wedge$
$transaccion\_cuenta(TDI, CUENTA) \longrightarrow$
$\exists\, CONTRATO(prestamo(TID, UID, FECHA, IMPORTE, CUENTA, CONTRATO)))$

$\forall\, UID, NOM\ (usuario(UID, NOM)\ \longrightarrow usuario(UID, NOM))$

Para el ejemplo, presento el siguiente:

TDI	UID	IMPORTE	FECHA	CUENTA	TIPO
1	1	10,2	12/10/1990	cuenta1	$"ES\_PRESTAMO"$
2	2	1232,12	25/09/2012	cuenta2	$"ES\_PAGO"$

TDI	CUENTA
1	cuenta1
2	cuenta2

UID	NOM
1	Karen Sanchez
2	Karen Carles

Y el destino queda así:

TDI	UID	IMPORTE	FECHA	CUENTA	CONTRATO
1	1	10,2	12/10/1990	cuenta1	contrato1

UID	NOM
1	Karen Sanchez
2	Karen Carles

5. PRÁCTICO TEMA 5, 6 Y 7

1.a. El fragmento es "guarded"porque la única variable es X y $phdStudent(X) \implies student(X)$ la contiene. Por otro lado no es "linear"porque existen otros átomos aparte de este último.

b. $chase(D, \Sigma_T) = D \cup \{...\}$

Por "$phdStudent(john)$".

$chase(D, \Sigma_T) = \{phdStudent(john)\} \cup \{student(john), supervisor(z1, john)\}$

Por "$phdStudent(X) \implies student(X)$".

$chase(D, \Sigma_T) = \{phdStudent(john)\} \cup \{student(john), ...\}$

Por "$phdStudent(X) \implies \exists Y supervisor(Y, X)$".

$chase(D, \Sigma_T) = \{phdStudent(john)\} \cup \{student(john), supervisor(z1, john)\}$

c. La respuesta de $Q() = \exists X supervisor(X, john)$ es $True$ ya que por el átomo "$phdStudent(X) \implies \exists Y supervisor(Y, X)$ɀ por "$phdStudent(john)$".

Por otro lado la respuesta a $Q(X) = supervisor(X, john)$ es $\{\}$. Ya que la base de datos D no nos provee más información que el hecho de que esta "$phdStudent(X)$".

3. a a. $F = \{A, R\}$ donde:
$A = \{A, B, C, D, E, F, G\}$
$R = \{(A, B), (A, F), (B, D), (D, C), (C, E), (E, F), (G, C)\}$

b. $\{B, F\}$ es un conjunto que es conflict free ya que ninguno de los dos miembros del mismo ataca al otro. Pero no es admisible porque tanto B como F son atacados por A y ninguno se puede defender de este.

c. $\{A, D, E\}$ ninguno se ataca con lo cual es conflict free, E es atacado por C, pero es defendido por D, D es atacado por B pero es defendido por A. A no es atacado por nadie. Con lo cual es admissible.

$\{G, E\}$ ninguno se ataca con lo que es conflict free, E es atacado por C pero defendido por G, nadie ataca a G. Con lo cual es admissible.