Especificación I

Nicolás D'Ippolito[1], Ariel Bendersky[1]

[1]Instituto de Investigaciones en Ciencias de la Computación, UBA-CONICET, Argentina

Algoritmos y Estructuras de Datos II
Segundo cuatrimestre de 2019

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠
- Diapos numeradas.

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠
- Diapos numeradas.
- Su NO pregunta SÍ molesta.

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠
- Diapos numeradas.
- Su NO pregunta SÍ molesta.
- Las siguientes cosas no son equivalentes (de a pares):

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠
- Diapos numeradas.
- Su NO pregunta SÍ molesta.
- Las siguientes cosas no son equivalentes (de a pares):
  1. Presenciar esta clase.

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠
- Diapos numeradas.
- Su NO pregunta SÍ molesta.
- Las siguientes cosas no son equivalentes (de a pares):
  1. Presenciar esta clase.
  2. Leer los apuntes.

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠
- Diapos numeradas.
- Su NO pregunta SÍ molesta.
- Las siguientes cosas no son equivalentes (de a pares):
  1. Presenciar esta clase.
  2. Leer los apuntes.
  3. Presenciar las clases prácticas sobre el tema.

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠
- Diapos numeradas.
- Su NO pregunta SÍ molesta.
- Las siguientes cosas no son equivalentes (de a pares):
  1. Presenciar esta clase.
  2. Leer los apuntes.
  3. Presenciar las clases prácticas sobre el tema.
  4. Hacer los ejercicios de la práctica.

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠
- Diapos numeradas.
- Su NO pregunta SÍ molesta.
- Las siguientes cosas no son equivalentes (de a pares):
  1. Presenciar esta clase.
  2. Leer los apuntes.
  3. Presenciar las clases prácticas sobre el tema.
  4. Hacer los ejercicios de la práctica.
- ¿Cómo sé si entedí los temas?

- Cosas importantes (tal vez no las únicas):
- Diapos numeradas.
- Su NO pregunta SÍ molesta.
- Las siguientes cosas no son equivalentes (de a pares):
  1. Presenciar esta clase.
  2. Leer los apuntes.
  3. Presenciar las clases prácticas sobre el tema.
  4. Hacer los ejercicios de la práctica.
- ¿Cómo sé si entedí los temas?
  - Los prácticos: si me salen los ejercicios (en un tiempo razonable).

(2) Algunas aclaraciones preliminares

- Cosas importantes (tal vez no las únicas): ⚠
- Diapos numeradas.
- Su NO pregunta SÍ molesta.
- Las siguientes cosas no son equivalentes (de a pares):
  1. Presenciar esta clase.
  2. Leer los apuntes.
  3. Presenciar las clases prácticas sobre el tema.
  4. Hacer los ejercicios de la práctica.
- ¿Cómo sé si entedí los temas?
  - Los prácticos: si me salen los ejercicios (en un tiempo razonable).
  - Los teóricos: si soy capaz de explicarlos con mis propias palabras.

(3) Contexto

- En esta materia vamos a estudiar algoritmos y estructuras de datos.

(3) Contexto

- En esta materia vamos a estudiar algoritmos y estructuras de datos.
- Recordemos qué es un algoritmo:

(3) Contexto

- En esta materia vamos a estudiar algoritmos y estructuras de datos.
- Recordemos qué es un algoritmo:
  - Secuencia ordenada y finita de pasos que nos llevan de un estado inicial a uno final.

(3) Contexto

- En esta materia vamos a estudiar algoritmos y estructuras de datos.
- Recordemos qué es un algoritmo:
  - Secuencia ordenada y finita de pasos que nos llevan de un estado inicial a uno final.
- Una de las cosas más importantes al estudiar algoritmos es considerarlos como una herramienta para la resolución de problemas (como opuesto a recitarlos), de manera tal de poder enfrentar un problema y poder combinarlos, modificarlos, etc.

(3) Contexto

- En esta materia vamos a estudiar algoritmos y estructuras de datos.
- Recordemos qué es un algoritmo:
  - Secuencia ordenada y finita de pasos que nos llevan de un estado inicial a uno final.
- Una de las cosas más importantes al estudiar algoritmos es considerarlos como una herramienta para la resolución de problemas (como opuesto a recitarlos), de manera tal de poder enfrentar un problema y poder combinarlos, modificarlos, etc.
- ¿Qué es resolver un problema ("genérico")?

(3) Contexto

- En esta materia vamos a estudiar algoritmos y estructuras de datos.
- Recordemos qué es un algoritmo:
  - Secuencia ordenada y finita de pasos que nos llevan de un estado inicial a uno final.
- Una de las cosas más importantes al estudiar algoritmos es considerarlos como una herramienta para la resolución de problemas (como opuesto a recitarlos), de manera tal de poder enfrentar un problema y poder combinarlos, modificarlos, etc.
- ¿Qué es resolver un problema ("genérico")?
  - Se trata de, dada su descripción, proponer un algoritmo que resuelva cualquier instancia del mismo.

(3) Contexto

- En esta materia vamos a estudiar algoritmos y estructuras de datos.
- Recordemos qué es un algoritmo:
  - Secuencia ordenada y finita de pasos que nos llevan de un estado inicial a uno final.
- Una de las cosas más importantes al estudiar algoritmos es considerarlos como una herramienta para la resolución de problemas (como opuesto a recitarlos), de manera tal de poder enfrentar un problema y poder combinarlos, modificarlos, etc.
- ¿Qué es resolver un problema ("genérico")?
  - Se trata de, dada su descripción, proponer un algoritmo que resuelva cualquier instancia del mismo.
- Veamos algunos ejemplos de resolución de problemas.

(4) Resolución de problemas

- Descripción del problema: “Dados dos números enteros, sumarlos”.

(4) Resolución de problemas

- Descripción del problema: "Dados dos números enteros, sumarlos".
- Algoritmo: `resultado:= a+b`

(5) Resolución de problemas (cont.)

- Descripción del problema: “Dados cuatro reales encontrar al entero más chico que esté por encima del mínimo cuadrado perfecto mayor que el mínimo de los dos primeros pero menor que la suma de los otros dos”.

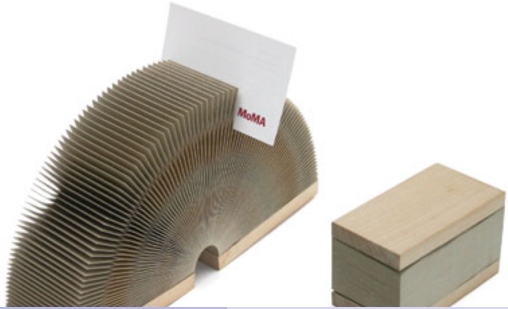
(5) Resolución de problemas (cont.)

- Descripción del problema: "Dados cuatro reales encontrar al entero más chico que esté por encima del mínimo cuadrado perfecto mayor que el mínimo de los dos primeros pero menor que la suma de los otros dos".
- Algoritmo: ¿? .

- Descripción del problema: “Dados cuatro reales encontrar al entero más chico que esté por encima del mínimo cuadrado perfecto mayor que el mínimo de los dos primeros pero menor que la suma de los otros dos”.
- Algoritmo: ¿? .
- Especifiquémoslo:
  problema ideaTrasnochada(a, b, c, d: float) = resultado: int {
  requiere $a > 0 \land b > 0$;
  asegura
  $z = min\{x | \sqrt{x} / int(\sqrt{x}) = 1 \land min(a, b) < x < c+d\} \land resultado = min\{w | w \geq z\}$;
  }

- Descripción del problema: "Dados cuatro reales encontrar al entero más chico que esté por encima del mínimo cuadrado perfecto mayor que el mínimo de los dos primeros pero menor que la suma de los otros dos".
- Algoritmo: ¿? .
- Especifiquémoslo:
  problema ideaTrasnochada(a, b, c, d: float) = resultado: int {
  requiere $a > 0 \land b > 0$;
  asegura
  $z = min\{x | \sqrt{x}/int(\sqrt{x}) = 1 \land min(a, b) < x < c+d\} \land resultado = min\{w | w \geq z\}$;
  }
- Notemos cómo al introducir el lenguaje matemático, con su precisión, se nos hace más fácil entender qué se nos pide. Presentar un algoritmo ahora no parece *tan* difícil.

- Descripción: "El dueño de un restaurant quiere asegurarse de que los pedidos de sus clientes sean atendidos con prolijidad. Los mozos llevan los pedidos hasta la cocina donde los colocan en un rotador de tarjetas (tipo rolodex). Cuando el cocinero se libera, saca la primera y prepara el plato allí indicado. El dueño quiere saber cuál es el próximo plato a preparar y cuántos pedidos atiende el cocinero cada día, y cuál fue el día con menos pedidos."



(7) Resolución de problemas (cont.)

- Algoritmo: ¿?

(7) Resolución de problemas (cont.)

- Algoritmo: ¿?
- Especificación: ¿?

(7) Resolución de problemas (cont.)

- Algoritmo: ¿?
- Especificación: ¿?
- No parece que el problema se pueda expresar directamente en un lenguaje matemático, pero está claro que necesitamos algo riguroso y formal.

(7) Resolución de problemas (cont.)

- Algoritmo: ¿?
- Especificación: ¿?
- No parece que el problema se pueda expresar directamente en un lenguaje matemático, pero está claro que necesitamos algo riguroso y formal.
- Una observación: con sólo mirar el enunciado vemos que se pretenden obtener resultados de varios tipos distintos: al menos naturales y platos. Además, hay varias operaciones.

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.
- Veamos qué tipo de cosas maneja el problema: hay platos, días y (al menos un) restaurant.

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.
- Veamos qué tipo de cosas maneja el problema: hay platos, días y (al menos un) restaurant.
- ¿Qué nos interesa saber de cada uno de estas "cosas"?

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.
- Veamos qué tipo de cosas maneja el problema: hay platos, días y (al menos un) restaurant.
- ¿Qué nos interesa saber de cada uno de estas "cosas"?
  - De los días,

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.
- Veamos qué tipo de cosas maneja el problema: hay platos, días y (al menos un) restaurant.
- ¿Qué nos interesa saber de cada uno de estas “cosas”?
  - De los días,

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.
- Veamos qué tipo de cosas maneja el problema: hay platos, días y (al menos un) restaurant.
- ¿Qué nos interesa saber de cada uno de estas "cosas"?
    - De los días, que pasan.

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.
- Veamos qué tipo de cosas maneja el problema: hay platos, días y (al menos un) restaurant.
- ¿Qué nos interesa saber de cada uno de estas "cosas"?
  - De los días, que pasan.
  - De los platos,

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.
- Veamos qué tipo de cosas maneja el problema: hay platos, días y (al menos un) restaurant.
- ¿Qué nos interesa saber de cada uno de estas “cosas”?
  - De los días, que pasan.
  - De los platos,

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.
- Veamos qué tipo de cosas maneja el problema: hay platos, días y (al menos un) restaurant.
- ¿Qué nos interesa saber de cada uno de estas "cosas"?
  - De los días, que pasan.
  - De los platos, casi nada, basta con poder diferenciarlos entre sí.

(8) Empecemos a modelar

- Vamos a modelarlo usando una herramienta nueva: tipos abstractos de datos.
- Veamos qué tipo de cosas maneja el problema: hay platos, días y (al menos un) restaurant.
- ¿Qué nos interesa saber de cada uno de estas “cosas”?
  - De los días, que pasan.
  - De los platos, casi nada, basta con poder diferenciarlos entre sí.
  - De los restaurantes, varias cosas, así que dejémoslos de lado por un momento.

Usemos renombres:

- **TAD** DÍA **ES** NAT

(9) Lo más sencillo

Usemos renombres:

- **TAD** DÍA **ES** NAT
- **TAD** PLATO **ES** STRING

(9) Lo más sencillo

Usemos renombres:

- **TAD** Día **ES** Nat
- **TAD** Plato **ES** String
- ¿Podríamos haber usado String directamente?

(9) Lo más sencillo

Usemos renombres:

- **TAD** Día **ES** Nat
- **TAD** Plato **ES** String
- ¿Podríamos haber usado String directamente?
- Pasemos ahora al restaurant...

TAD RESTAURANT

operaciones

inaugurar : $\longrightarrow$ restaurant

cant_platos_pendientes : restaurant $\longrightarrow$ nat

próximo_pedido : restaurant r $\longrightarrow$ plato {cant_platos_pendientes(r)>0}

preparar_plato : restaurant r $\longrightarrow$ restaurant
{cant_platos_pendientes(r)>0}

tomar_pedido : restaurant $\times$ plato $\longrightarrow$ restaurant

nuevo_día : restaurant $\longrightarrow$ restaurant

día_actual : restaurant $\longrightarrow$ día

platos_por_día : restaurant r $\times$ día d $\longrightarrow$ nat {d $\leq$ día_actual(r)}

día_menos_pedidos : restaurant $\longrightarrow$ día

Fin TAD

- Lo que vimos es la *signatura*.

- Lo que vimos es la *signatura*.
  - Nos dice qué operaciones tiene el tipo, con qué parámetros y qué devuelven.

(11) TADs

- Lo que vimos es la *signatura*.
  - Nos dice qué operaciones tiene el tipo, con qué parámetros y qué devuelven.
  - ⚠ Las operaciones de los TADs son *funciones totales*. Ie, funciones que están definidas para cada valor del dominio. Por eso, en casos como `preparar_plato()` debemos restringir el dominio.

- Lo que vimos es la *signatura*.
  - Nos dice qué operaciones tiene el tipo, con qué parámetros y qué devuelven.
  - ⚠ Las operaciones de los TADs son *funciones totales*. Ie, funciones que están definidas para cada valor del dominio. Por eso, en casos como `preparar_plato()` debemos restringir el dominio.

⚠ Necesitamos darle comportamiento a las operaciones. En los TADs usaremos para eso axiomas.

(11) TADs

- Lo que vimos es la *signatura*.
  - Nos dice qué operaciones tiene el tipo, con qué parámetros y qué devuelven.
  - ⚠ Las operaciones de los TADs son *funciones totales*. Ie, funciones que están definidas para cada valor del dominio. Por eso, en casos como `preparar_plato()` debemos restringir el dominio.
- ⚠ Necesitamos darle comportamiento a las operaciones. En los TADs usaremos para eso axiomas.
  - Algunos serán ecuacionales: día_actual(nuevo_día(r)) $\equiv$ día_actual(r) $+1$

- Lo que vimos es la *signatura*.
  - Nos dice qué operaciones tiene el tipo, con qué parámetros y qué devuelven.
  - ⚠ Las operaciones de los TADs son *funciones totales*. Ie, funciones que están definidas para cada valor del dominio. Por eso, en casos como `preparar_plato()` debemos restringir el dominio.
- ⚠ Necesitamos darle comportamiento a las operaciones. En los TADs usaremos para eso axiomas.
  - Algunos serán ecuacionales: $\text{día\_actual}(\text{nuevo\_día}(\text{r})) \equiv \text{día\_actual}(\text{r}) + 1$
  - Expresan reescritura de términos.

- Lo que vimos es la *signatura*.
  - Nos dice qué operaciones tiene el tipo, con qué parámetros y qué devuelven.
  - ⚠ Las operaciones de los TADs son *funciones totales*. Ie, funciones que están definidas para cada valor del dominio. Por eso, en casos como `preparar_plato()` debemos restringir el dominio.
- ⚠ Necesitamos darle comportamiento a las operaciones. En los TADs usaremos para eso axiomas.
  - Algunos serán ecuacionales: día_actual(nuevo_día(r)) $\equiv$ día_actual(r) $+1$
  - Expresan reescritura de términos.
  - ¿Qué es un término? ¿Qué no lo es?

axiomas

día_actual(inaugurar()) $\equiv$ 0
($\forall$ r : restaurant) día_actual(nuevo_día(r)) $\equiv$ día_actual(r)+1
($\forall$ r : restaurant) ($\forall$ p : plato) día_actual(tomar_pedido(r, p)) $\equiv$ día_actual(r)

...

cant_platos_pendientes(inaugurar()) $\equiv$ 0
($\forall$ r : restaurant) ($\forall$ p : plato) cant_platos_pendientes(tomar_pedido(r, p)) $\equiv$
cant_platos_pendientes(r)+1
($\forall$ r : restaurant) cant_platos_pendientes(preparar_plato(r)) $\equiv$
cant_platos_pendientes(r)$-$1

(13) Los axiomas (cont.)

(No escribo más los $\forall$, pero siguen estando sobre las variables libres.)

...

próximo_pedido(r) $\equiv$ ult(secuencia_de_pedidos(r))
secuencia_de_pedidos(inaugurar()) $\equiv$ $<>$
secuencia_de_pedidos(tomar_pedido(r, p)) $\equiv$ secuencia_de_pedidos(r) $\bullet$ p
secuencia_de_pedidos(preparar_plato(r)) $\equiv$ com(secuencia_de_pedidos(r))

. . .

platos_por_día(d, inaugurar()) $\equiv$ 0
platos_por_día(d, tomar_pedido(r, p)) $\equiv$ platos_por_día(d, r)
platos_por_día(d, preparar_plato(r)) $\equiv$ **if** día_actual(r) $\equiv$ d **then**
    platos_por_día(d, r)+1
else
    platos_por_día(d, r)
fi
platos_por_día(d, nuevo_día(r)) $\equiv$ **if** día_actual(r)+1 $\equiv$ d **then**
    0
else
    platos_por_día(d, r)
fi

(15) Los axiomas (cont.)

...

$$(\forall d' : \text{día})\ 0 \leq d' \leq \text{día\_actual(r)} \ \Rightarrow$$
$$\text{platos\_por\_día(r, día\_menos\_pedidos(r))} \leq \text{platos\_por\_día(r, d')}$$

(16) ¡Qué ganas de jorobar!

- Pseee..., esto ya lo vimos en Algo I.
- ¡Grande, JPG! ¡Volvé, Marenco!
- El lenguaje te lo cambian a propósito, para joderte.

(16) ¡Qué ganas de jorobar!

- Pseee..., esto ya lo vimos en Algo I.
  - Parcialmente cierto, vieron sólo una parte del asunto.
- ¡Grande, JPG! ¡Volvé, Marenco!
- El lenguaje te lo cambian a propósito, para joderte.

(16) ¡Qué ganas de jorobar!

- Pseee..., esto ya lo vimos en Algo I.
  - Parcialmente cierto, vieron sólo una parte del asunto.
- ¡Grande, JPG! ¡Volvé, Marenco!
  - Absolutamente cierto, son dos fenónemos.

- El lenguaje te lo cambian a propósito, para joderte.

(16) ¡Qué ganas de jorobar!

- Pseee..., esto ya lo vimos en Algo I.
  - Parcialmente cierto, vieron sólo una parte del asunto.
- ¡Grande, JPG! ¡Volvé, Marenco!
  - Absolutamente cierto, son dos fenónemos.

- El lenguaje te lo cambian a propósito, para joderte.
  - Absolutamente falso, el lenguaje axiomático permite estudiar otras formas de demostración muy útiles, como la *inducción estructural*.

(17) ¿Qué es un TAD?

- ¿Qué es un TAD?

- ¿Qué es un TAD?

(17) ¿Qué es un TAD?

- ¿Qué es un TAD? Es una pregunta multifacética.

(17) ¿Qué es un TAD?

- ¿Qué es un TAD? Es una pregunta multifacética.
- Desde el punto de vista formal: es una herramienta lógico/matemática. Eso es bueno porque nos da la rigurosidad que necesitábamos para entender claramente qué hay que hacer.

- ¿Qué es un TAD? Es una pregunta multifacética.
- Desde el punto de vista formal: es una herramienta lógico/matemática. Eso es bueno porque nos da la rigurosidad que necesitábamos para entender claramente qué hay que hacer.
    - A título informativo: cada TAD constituye una *teoría de primer orden con igualdad*.

(17) ¿Qué es un TAD?

- ¿Qué es un TAD? Es una pregunta multifacética.
- Desde el punto de vista formal: es una herramienta lógico/matemática. Eso es bueno porque nos da la rigurosidad que necesitábamos para entender claramente qué hay que hacer.
  - A título informativo: cada TAD constituye una *teoría de primer orden con igualdad*.
- Desde el punto de vista práctico: es una herramienta poderosa y flexible.

(17) ¿Qué es un TAD?

- ¿Qué es un TAD? Es una pregunta multifacética.
- Desde el punto de vista formal: es una herramienta lógico/matemática. Eso es bueno porque nos da la rigurosidad que necesitábamos para entender claramente qué hay que hacer.
  - A título informativo: cada TAD constituye una *teoría de primer orden con igualdad*.
- Desde el punto de vista práctico: es una herramienta poderosa y flexible.
- Desde el punto de vista histórico: uno de los primeros intentos por abordar este problema.

(17) ¿Qué es un TAD?

- ¿Qué es un TAD? Es una pregunta multifacética.
- Desde el punto de vista formal: es una herramienta lógico/matemática. Eso es bueno porque nos da la rigurosidad que necesitábamos para entender claramente qué hay que hacer.
    - A título informativo: cada TAD constituye una *teoría de primer orden con igualdad*.
- Desde el punto de vista práctico: es una herramienta poderosa y flexible.
- Desde el punto de vista histórico: uno de los primeros intentos por abordar este problema.
- Desde el punto de vista pedagógico: un primer paso muy importante. Al dominar los tipos abstractos de datos se aprenden gran parte de los conceptos del mundo de la orientación a objetos.

(18) Paréntesis lógico

- Lógica trivaluada: *true*, *false*, $\perp$.

(18) Paréntesis lógico

- Lógica trivaluada: *true*, *false*, $\perp$.
- Vamos a usar la versión $_L$ de los conectivos lógicos.

(18) Paréntesis lógico

- Lógica trivaluada: *true*, *false*, $\perp$.
- Vamos a usar la versión $_L$ de los conectivos lógicos.
- Variables ligadas: $((\forall x)\ P(x)) \wedge ((\exists y)\ Q(y)) \equiv ((\forall x)\ P(x)) \wedge ((\exists x)\ Q(x))$

(18) Paréntesis lógico

- Lógica trivaluada: *true*, *false*, $\perp$.
- Vamos a usar la versión $_L$ de los conectivos lógicos.
- Variables ligadas: $((\forall x)\ P(x)) \wedge ((\exists y)\ Q(y)) \equiv ((\forall x)\ P(x)) \wedge ((\exists x)\ Q(x))$
- Cuantificadores: (CUANT var: género) P(var)

(18) Paréntesis lógico

- Lógica trivaluada: *true*, *false*, $\perp$.
- Vamos a usar la versión $_L$ de los conectivos lógicos.
- Variables ligadas: $((\forall x)\ P(x)) \wedge ((\exists y)\ Q(y)) \equiv ((\forall x)\ P(x)) \wedge ((\exists x)\ Q(x))$
- Cuantificadores: (CUANT var: género) P(var)
- $(\exists x : nat)\ P(x) \approx P(0) \vee P(1) \vee \ldots$

(18) Paréntesis lógico

- Lógica trivaluada: *true*, *false*, $\perp$.
- Vamos a usar la versión $_L$ de los conectivos lógicos.
- Variables ligadas: $((\forall x)\ P(x)) \wedge ((\exists y)\ Q(y)) \equiv ((\forall x)\ P(x)) \wedge ((\exists x)\ Q(x))$
- Cuantificadores: (CUANT var: género) P(var)
- $(\exists x : nat)\ P(x) \approx P(0) \vee P(1) \vee \ldots$
- $(\forall x)\ P(x) \equiv \neg((\exists x)\ \neg P(x))$ (ie, "todos los x satisfacen P" $\equiv$ "no hay ningún x que no cumpla P")

(18) Paréntesis lógico

- Lógica trivaluada: *true*, *false*, $\perp$.
- Vamos a usar la versión $_L$ de los conectivos lógicos.
- Variables ligadas: $((\forall x)\ P(x)) \wedge ((\exists y)\ Q(y)) \equiv ((\forall x)\ P(x)) \wedge ((\exists x)\ Q(x))$
- Cuantificadores: (CUANT var: género) P(var)
- $(\exists x : nat)\ P(x) \approx P(0) \vee P(1) \vee \ldots$
- $(\forall x)\ P(x) \equiv \neg((\exists x)\ \neg P(x))$ (ie, "todos los x satisfacen P" $\equiv$ "no hay ningún x que no cumpla P")
- "Expandiendo" el existencial y aplicando De Morgan: $\approx P(0) \wedge P(1) \wedge \ldots$

(18) Paréntesis lógico

- Lógica trivaluada: *true*, *false*, $\perp$.
- Vamos a usar la versión $_L$ de los conectivos lógicos.
- Variables ligadas: $((\forall x)\ P(x)) \wedge ((\exists y)\ Q(y)) \equiv ((\forall x)\ P(x)) \wedge ((\exists x)\ Q(x))$
- Cuantificadores: (CUANT var: género) P(var)
- $(\exists x : nat)\ P(x) \approx P(0) \vee P(1) \vee \ldots$
- $(\forall x)\ P(x) \equiv \neg((\exists x)\ \neg P(x))$ (ie, "todos los x satisfacen P" $\equiv$ "no hay ningún x que no cumpla P")
- "Expandiendo" el existencial y aplicando De Morgan: $\approx P(0) \wedge P(1) \wedge \ldots$

⚠ Las expansiones son una "forma de decirlo", pero rigurosamente no es así: pensar en indefiniciones e infinitud.

(19) Paréntesis lógico (cont.)

- En general los vamos a usar con rangos. Eg, $(\forall x : \mathit{nat})\ (1 \leq x) \Rightarrow_{\mathrm{L}} x/x = 1$

(19) Paréntesis lógico (cont.)

- En general los vamos a usar con rangos. Eg, $(\forall x : \mathit{nat})\ (1 \leq x) \Rightarrow_{\mathrm{L}} x/x = 1$
- Es decir, vamos a escribir: $(\forall x : \text{género})\ (R(x) \Rightarrow_{\mathrm{L}} P(x))$, donde R nos dice cuáles son los x sobre los que nos interesa el predicado P.

(19) Paréntesis lógico (cont.)

- En general los vamos a usar con rangos. Eg, $(\forall x : nat)\ (1 \leq x) \Rightarrow_{\mathrm{L}} x/x = 1$
- Es decir, vamos a escribir: $(\forall x : \text{género})\ (R(x) \Rightarrow_{\mathrm{L}} P(x))$, donde R nos dice cuáles son los x sobre los que nos interesa el predicado P.
- ¿Qué pasa con el existencial?

(19) Paréntesis lógico (cont.)

- En general los vamos a usar con rangos. Eg, $(\forall x : \mathit{nat})\ (1 \leq x) \Rightarrow_{\mathrm{L}} x/x = 1$
- Es decir, vamos a escribir: $(\forall x : \text{género})\ (R(x) \Rightarrow_{\mathrm{L}} P(x))$, donde R nos dice cuáles son los x sobre los que nos interesa el predicado P.
- ¿Qué pasa con el existencial?

(19) Paréntesis lógico (cont.)

- En general los vamos a usar con rangos. Eg, $(\forall x : \mathit{nat})\ (1 \leq x) \Rightarrow_{\mathrm{L}} x/x = 1$
- Es decir, vamos a escribir: $(\forall x : \text{género})\ (R(x) \Rightarrow_{\mathrm{L}} P(x))$, donde R nos dice cuáles son los x sobre los que nos interesa el predicado P.
- ¿Qué pasa con el existencial? $(\exists x : \text{género})\ (R(x) \wedge_{\mathrm{L}} P(x))$.

- En general los vamos a usar con rangos. Eg, $(\forall x : \mathit{nat})\ (1 \leq x) \Rightarrow_{\mathrm{L}} x/x = 1$
- Es decir, vamos a escribir: $(\forall x : \text{género})\ (R(x) \Rightarrow_{\mathrm{L}} P(x))$, donde R nos dice cuáles son los x sobre los que nos interesa el predicado P.
- ¿Qué pasa con el existencial? $(\exists x : \text{género})\ (R(x) \wedge_{\mathrm{L}} P(x))$.

⚠ Notar: si el rango es vacío ($R(x)$ no vale para ningún x) el $\forall$ da verdadero. En el caso de un $\exists$, da falso (pensar en las expansiones de la transparencia anterior).

(20) Tarea para el hogar

- Existen naturales pares menores que 33.

(20) Tarea para el hogar

- Existen naturales pares menores que 33.
- Todos las secuencias de longitud par se pueden escribir como la concatenación de otras dos secuencias.

(20) Tarea para el hogar

- Existen naturales pares menores que 33.
- Todos las secuencias de longitud par se pueden escribir como la concatenación de otras dos secuencias.
- Todos los números primos tienen un elemento mayor y otro menor.

(20) Tarea para el hogar

- Existen naturales pares menores que 33.
- Todos las secuencias de longitud par se pueden escribir como la concatenación de otras dos secuencias.
- Todos los números primos tienen un elemento mayor y otro menor.
- Cada secuencia de naturales puede ser extendida en un elemento en tantas secuencias como naturales.

(20) Tarea para el hogar

- Existen naturales pares menores que 33.
- Todos las secuencias de longitud par se pueden escribir como la concatenación de otras dos secuencias.
- Todos los números primos tienen un elemento mayor y otro menor.
- Cada secuencia de naturales puede ser extendida en un elemento en tantas secuencias como naturales.
- Todo Z_n tiene su neutro aditivo.

(20) Tarea para el hogar

- Existen naturales pares menores que 33.
- Todos las secuencias de longitud par se pueden escribir como la concatenación de otras dos secuencias.
- Todos los números primos tienen un elemento mayor y otro menor.
- Cada secuencia de naturales puede ser extendida en un elemento en tantas secuencias como naturales.
- Todo Z_n tiene su neutro aditivo.
- Hay un neutro aditivo e para todo Z_n.

(21) Dónde estamos parados

- Queremos resolver problemas.

(21) Dónde estamos parados

- Queremos resolver problemas.
- Primero tenemos que expresar con claridad qué es lo que queremos resolver. En particular, usaremos un formalismo matemático.

(21) Dónde estamos parados

- Queremos resolver problemas.
- Primero tenemos que expresar con claridad qué es lo que queremos resolver. En particular, usaremos un formalismo matemático.
- Vimos que a medida que se complican los problemas, más poder necesitamos en nuestro formalismo y para eso introducimos los TADs.

(21) Dónde estamos parados

- Queremos resolver problemas.
- Primero tenemos que expresar con claridad qué es lo que queremos resolver. En particular, usaremos un formalismo matemático.
- Vimos que a medida que se complican los problemas, más poder necesitamos en nuestro formalismo y para eso introducimos los TADs.
- Planteamos el caso del restaurant como un ejemplo, como para ir "tomándole el gusto".

(21) Dónde estamos parados

- Queremos resolver problemas.
- Primero tenemos que expresar con claridad qué es lo que queremos resolver. En particular, usaremos un formalismo matemático.
- Vimos que a medida que se complican los problemas, más poder necesitamos en nuestro formalismo y para eso introducimos los TADs.
- Planteamos el caso del restaurant como un ejemplo, como para ir "tomándole el gusto".
- Vimos que necesitábamos manejar algunos conceptos de lógica que conocíamos de Álgebra I, pero que tal vez teníamos medio oxidados.

(21) Dónde estamos parados

- Queremos resolver problemas.
- Primero tenemos que expresar con claridad qué es lo que queremos resolver. En particular, usaremos un formalismo matemático.
- Vimos que a medida que se complican los problemas, más poder necesitamos en nuestro formalismo y para eso introducimos los TADs.
- Planteamos el caso del restaurant como un ejemplo, como para ir "tomándole el gusto".
- Vimos que necesitábamos manejar algunos conceptos de lógica que conocíamos de Álgebra I, pero que tal vez teníamos medio oxidados.
- Vamos a ver, ahora con más detenimiento, el lenguaje de los TADs, desde el principio.

(22) El génesis

- Una de las ventajas de la teoría de los tipos abstractos de datos es que no requiere de tipos primitivos que deban definirse por fuera de la misma.

(22) El génesis

- Una de las ventajas de la teoría de los tipos abstractos de datos es que no requiere de tipos primitivos que deban definirse por fuera de la misma.
- Veamos cómo se definen los valores booleanos y los naturales.

(22) El génesis

- Una de las ventajas de la teoría de los tipos abstractos de datos es que no requiere de tipos primitivos que deban definirse por fuera de la misma.
- Veamos cómo se definen los valores booleanos y los naturales.

⚠ El resto de los tipos básicos están en el apunte de tipos básicos, en la página de la materia. Deben leerlos de ahí.

(22) El génesis

- Una de las ventajas de la teoría de los tipos abstractos de datos es que no requiere de tipos primitivos que deban definirse por fuera de la misma.
- Veamos cómo se definen los valores booleanos y los naturales.

⚠ El resto de los tipos básicos están en el apunte de tipos básicos, en la página de la materia. Deben leerlos de ahí.

⚠ Lo mismo vale para los detalles de los tipos que presento hoy. No los voy a mostrar completos.

TAD Bool

géneros bool

exporta bool, generadores, $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\vee_{\mathrm{L}}$, $\wedge_{\mathrm{L}}$, $\Rightarrow_{\mathrm{L}}$

generadores

$true : \longrightarrow \text{bool}$

$false : \longrightarrow \text{bool}$

otras operaciones

$\neg\bullet : \text{bool} \longrightarrow \text{bool}$

$\bullet \vee \bullet : \text{bool} \times \text{bool} \longrightarrow \text{bool}$

$\bullet \wedge \bullet : \text{bool} \times \text{bool} \longrightarrow \text{bool}$

$\bullet \Rightarrow \bullet : \text{bool} \times \text{bool} \longrightarrow \text{bool}$

$\bullet \vee_{\mathrm{L}} \bullet : \text{bool} \times \text{bool} \longrightarrow \text{bool}$

$\bullet \wedge_{\mathrm{L}} \bullet : \text{bool} \times \text{bool} \longrightarrow \text{bool}$

$\bullet \Rightarrow_{\mathrm{L}} \bullet : \text{bool} \times \text{bool} \longrightarrow \text{bool}$

axiomas

$$\neg \mathit{true} \equiv \mathit{false}$$
$$\neg \mathit{false} \equiv \mathit{true}$$
$$(\forall\ x : \text{bool})\quad \mathit{true} \vee x \equiv \mathit{true}$$
$$(\forall\ x : \text{bool})\quad \mathit{false} \vee x \equiv x$$
$$(\forall\ x : \text{bool})\quad \mathit{true} \wedge x \equiv x$$
$$(\forall\ x : \text{bool})\quad \mathit{false} \wedge x \equiv \mathit{false}$$
$$(\forall\ x, y : \text{bool})\quad x \Rightarrow y \equiv \neg x \vee y$$

axiomas

$(\forall\ x : \mathsf{bool})\ \ x \wedge_{\mathrm{L}} y \ \equiv\ \textbf{if}\ \mathrm{x}\ \ \textbf{then}\ \ \mathrm{y}\ \ \textbf{else}\ \ \mathsf{false}\ \ \textbf{fi}$

$(\forall\ x : \mathsf{bool})\ \ x \vee_{\mathrm{L}} y \ \equiv\ \textbf{if}\ \mathrm{x}\ \ \textbf{then}\ \ \mathrm{true}\ \ \textbf{else}\ \ \mathrm{y}\ \ \textbf{fi}$

$(\forall\ x : \mathsf{bool})\ \ x \Rightarrow_{\mathrm{L}} y \ \equiv\ \neg x \vee_{\mathrm{L}} y$

(26) Secciones de un TAD

- **Géneros**. Los géneros (en general va a haber sólo uno, pero podrían ser más) son el nombre que recibe el conjunto de valores del tipo.

(26) Secciones de un TAD

- **Géneros**. Los géneros (en general va a haber sólo uno, pero podrían ser más) son el nombre que recibe el conjunto de valores del tipo.
- **Usa**. Inclusión de los genéros y operaciones exportadas de los tipos mencionados allí.

- **Géneros**. Los géneros (en general va a haber sólo uno, pero podrían ser más) son el nombre que recibe el conjunto de valores del tipo.
- **Usa**. Inclusión de los genéros y operaciones exportadas de los tipos mencionados allí.
- **Exporta**. Qué operaciones y géneros se dejan a disposición de los usuarios del tipo.

(26) Secciones de un TAD

- **Géneros**. Los géneros (en general va a haber sólo uno, pero podrían ser más) son el nombre que recibe el conjunto de valores del tipo.
- **Usa**. Inclusión de los genéros y operaciones exportadas de los tipos mencionados allí.
- **Exporta**. Qué operaciones y géneros se dejan a disposición de los usuarios del tipo.
- **Generadores.** Son operaciones que permiten construir valores del tipo. Un conjunto de generadores está bien armado si una combinación de ellos permite construir cualquier instancia posible del tipo.

(26) Secciones de un TAD

- **Géneros**. Los géneros (en general va a haber sólo uno, pero podrían ser más) son el nombre que recibe el conjunto de valores del tipo.
- **Usa**. Inclusión de los genéros y operaciones exportadas de los tipos mencionados allí.
- **Exporta**. Qué operaciones y géneros se dejan a disposición de los usuarios del tipo.
- **Generadores.** Son operaciones que permiten construir valores del tipo. Un conjunto de generadores está bien armado si una combinación de ellos permite construir cualquier instancia posible del tipo.
- **Observadores**. Son aquellas operaciones que nos permiten, utilizadas en conjunto, diferenciar instancias del tipo.

(26) Secciones de un TAD

- **Géneros**. Los géneros (en general va a haber sólo uno, pero podrían ser más) son el nombre que recibe el conjunto de valores del tipo.
- **Usa**. Inclusión de los genéros y operaciones exportadas de los tipos mencionados allí.
- **Exporta**. Qué operaciones y géneros se dejan a disposición de los usuarios del tipo.
- **Generadores.** Son operaciones que permiten construir valores del tipo. Un conjunto de generadores está bien armado si una combinación de ellos permite construir cualquier instancia posible del tipo.
- **Observadores**. Son aquellas operaciones que nos permiten, utilizadas en conjunto, diferenciar instancias del tipo.
- **Axiomas**. Son las reglas que nos explican el comportamiento de las funciones.

TAD Nat

géneros nat

exporta nat, generadores, observadores, $+$, $-$, $=$, $<$

usa Bool

observadores básicos

$\bullet =0? \ : \ \text{nat} \longrightarrow \text{bool}$

$\text{pred} \ : \ \text{nat } n \longrightarrow \text{nat} \qquad \{\neg(n = 0?)\}$

generadores

$0 \ : \ \longrightarrow \text{nat}$

$\text{suc} \ : \ \text{nat} \longrightarrow \text{nat}$

otras operaciones

$\bullet + \bullet \ : \ \text{nat} \times \text{nat} \longrightarrow \text{nat}$

$\bullet - \bullet \ : \ \text{nat } n \times \text{nat } m \longrightarrow \text{nat} \qquad \{\neg(n < m)\}$

$\bullet = \bullet \ : \ \text{nat} \times \text{nat} \longrightarrow \text{bool}$

$\bullet < \bullet \ : \ \text{nat} \times \text{nat} \longrightarrow \text{bool}$

Fin TAD

axiomas

$(\forall\ n, m : \text{nat})$

$0{=}0? \equiv \textit{true}$

$\text{suc}(n){=}\ 0? \equiv \textit{false}$

$\text{pred}(\text{suc}(n)) \equiv n$

$n + m \equiv \textbf{if } m{=}0? \textbf{ then } n \textbf{ else } \text{suc}(n{+}\text{pred}(m)) \textbf{ fi}$

$n = m \equiv (n{=}0? \equiv m{=}0?) \land_{\text{L}} (\neg(m{=}0?) \Rightarrow_{\text{L}} (\text{pred}(n) = \text{pred}(m)))$

$n < m \equiv \textbf{if } m{=}0? \textbf{ then}$

$\quad \textit{false}$

$\textbf{else}$

$\quad \textbf{if } \text{n}{=}0? \textbf{ then } \textit{true} \textbf{ else } \text{pred(n)}{<}\text{pred(m)} \textbf{ fi}$

$\textbf{fi}$

(29) Repaso de lo visto y perspectiva

- Vimos:

(29) Repaso de lo visto y perspectiva

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.

(29) Repaso de lo visto y perspectiva

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.
  - La conveniencia de hacerlo de manera formal.

(29) Repaso de lo visto y perspectiva

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.
  - La conveniencia de hacerlo de manera formal.
  - Una introducción a los TADs.

(29) Repaso de lo visto y perspectiva

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.
  - La conveniencia de hacerlo de manera formal.
  - Una introducción a los TADs.
  - El uso de cuantificadores.

(29) Repaso de lo visto y perspectiva

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.
  - La conveniencia de hacerlo de manera formal.
  - Una introducción a los TADs.
  - El uso de cuantificadores.
  - Algunos tipos básicos: BOOL, NAT.

(29) Repaso de lo visto y perspectiva

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.
  - La conveniencia de hacerlo de manera formal.
  - Una introducción a los TADs.
  - El uso de cuantificadores.
  - Algunos tipos básicos: $\textsc{Bool}$, $\textsc{Nat}$.
- Se viene

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.
  - La conveniencia de hacerlo de manera formal.
  - Una introducción a los TADs.
  - El uso de cuantificadores.
  - Algunos tipos básicos: $\textsc{Bool}$, $\textsc{Nat}$.
- Se viene
  - Más tipos básicos.

(29) Repaso de lo visto y perspectiva

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.
  - La conveniencia de hacerlo de manera formal.
  - Una introducción a los TADs.
  - El uso de cuantificadores.
  - Algunos tipos básicos: $\textsc{Bool}$, $\textsc{Nat}$.
- Se viene
  - Más tipos básicos.
  - Entender mejor la forma de escribir los TADs.

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.
  - La conveniencia de hacerlo de manera formal.
  - Una introducción a los TADs.
  - El uso de cuantificadores.
  - Algunos tipos básicos: BOOL, NAT.
- Se viene
  - Más tipos básicos.
  - Entender mejor la forma de escribir los TADs.
  - Empezar a axiomatizar.

(29) Repaso de lo visto y perspectiva

- Vimos:
  - La necesidad de especificar como un paso previo a la resolución de problemas.
  - La conveniencia de hacerlo de manera formal.
  - Una introducción a los TADs.
  - El uso de cuantificadores.
  - Algunos tipos básicos: Bool, Nat.
- Se viene
  - Más tipos básicos.
  - Entender mejor la forma de escribir los TADs.
  - Empezar a axiomatizar.
  - Empezar a modelar.

(30) Tarea

⚠ Leer el apunte de TADs básicos.

- ⚠ Leer el apunte de TADs básicos.
- ⚠ Empezar a hacer la práctica 1, parte 1.

(30) Tarea

⚠ Leer el apunte de TADs básicos.

⚠ Empezar a hacer la práctica 1, parte 1.

⚠ Hacer los ejercicios de cuantificación.