Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática
Paradigmas de Programación

Paradigmas de Programación

Proyecto semestral de laboratorio

Versión Preliminar - actualizada al 08/11/2023

Laboratorio 2 (Paradigma Lógico - Lenguaje Prolog)

(Cambios menores pueden incorporarse en futuras versiones a fin de aclarar o corregir errores)

(Sus dudas las puede expresar en este mismo enunciado, incluso puede responder a preguntas de compañeros en caso de que conozca la respuesta)

Enunciado General: Procure consultar los aspectos generales del proyecto de laboratorio en el documento general.

Fecha de Entrega: Ver calendario clase a clase donde se señala el hito

Objetivo del laboratorio: Aplicar conceptos del paradigma de programación lógico usando el lenguaje de programación Prolog en la resolución de un problema acotado.

Resultado esperado: Programa para realizar una simulación de un chatbot.

Profesor responsable: Víctor Flores (al hacer consultas en este documento, procurar hacer la mención a **@Víctor Flores Sánchez** (victor.floress@usach.cl) para que las notificaciones de sus consultas lleguen al profesor correspondiente)

Recomendaciones: El laboratorio está diseñado como un conjunto de ejercicios a abordar bajo cada paradigma. En este sentido, el desarrollo del laboratorio constituye un espacio para practicar y prepararse además para la evaluación de cátedra del correspondiente paradigma. Por tanto, se recomienda incorporar en sus hábitos de estudio/trabajo el desarrollo de las funcionalidades de forma diaria. En total el laboratorio 1 lista N funcionalidades a cubrir. Para alcanzar la nota de aprobación, se deben cubrir las M primeras y para la nota máxima se pueden cubrir Q más. Procure destinar tiempo para analizar y hacer una propuesta de diseño para el laboratorio completo antes de proceder a la implementación de la solución. No es necesario que sus predicados implementen comprobación de tipo, esto es opcional.

Requerimientos No Funcionales. Algunos son ineludibles/obligatorios, esto quiere decir que al no cumplir con dicho requerimiento, su proyecto será evaluado con la nota mínima.

1. **(obligatorio) Autoevaluación:** Incluir autoevaluación de cada uno de los requerimientos funcionales solicitados. El objetivo de esta autoevaluación es que usted pueda anticiparse a una nota tentativa al momento de la entrega del laboratorio para posteriormente, en caso de que su calificación sea inferior a 4.0, pueda proceder oportunamente a realizar mejoras en su laboratorio a través del comodín correspondiente.

2. **(obligatorio) Lenguaje:** La implementación debe ser en el lenguaje de programación Prolog en base a una programación principalmente declarativa - lógica.

3. **(obligatorio) Versión:** Usar swi-prolog versión 9.0 o superior.

4. **(obligatorio) Standard:** Se deben utilizar predicados estándar del lenguaje. No emplear bibliotecas externas.

5. **(1 pts) Documentación:** Todos las predicados deben estar debidamente comentados. Indicando descripción del predicado, tipo de algoritmo/estrategia empleado (ej: fuerza bruta, backtracking, si aplica) argumentos de entrada y de salida.

6. **(1 pts) Organización:** Estructurar su código en archivos independientes. Un archivo para cada TDA implementado y uno para el programa principal donde se dispongan sólo los predicados requeridos en el apartado de requerimientos funcionales.

7. **(2.5 pts) Historial:** Historial de trabajo en Github tomando en consideración la evolución en el desarrollo de su proyecto en distintas etapas. Se requieren **al menos 10 commits** distribuidos en un periodo de tiempo **mayor o igual a 2 semanas (no espere a terminar la materia para empezar a trabajar en el laboratorio. Puede hacer pequeños incrementos conforme avance el curso)**. Los criterios que se consideran en la evaluación de este ítem son: fecha primer commit, fecha último commit, total commits y máximo de commits diarios. A modo de ejemplo (y solo como una referencia), si hace todos los commits el día antes de la entrega del proyecto, este ítem tendrá 0 pts. De manera similar, si hace dos commits dos semanas antes de la entrega final y el resto los concentra en los últimos dos días, tendrá una evaluación del 25% para este ítem (0.375 pts). Por el contrario, si demuestra constancia en los commits (con aportes claros entre uno y otro) a lo largo del periodo evaluado, este ítem será evaluado con el total del puntaje.

8. **(obligatorio) Script de pruebas (pruebas_RUT_Apellidos.pl):** Incluir como parte de su entregable un archivo independiente al código donde muestre de forma completa, con la documentación correspondiente, el funcionamiento de su programa. Este archivo será similar al script de prueba proporcionado al final de este documento. Este archivo debe incluir los ejemplos provistos en el script de prueba de este enunciado además de **3 ejemplos** por cada una de los predicados requeridos. **Solo se revisarán proyectos que incluyan este archivo.**

9. **(obligatorio) Prerrequisitos:** Para cada funcionalidad se establecen prerrequisitos. Estos deben ser cumplidos para que se proceda con la evaluación de la funcionalidad implementada. Ej: Para evaluar el predicado *login*, debe estar implementado el predicado *register*.

Requerimientos Funcionales. **Para que el requerimiento sea evaluado, DEBE cumplir con el prerrequisito de evaluación y requisito de implementación. En caso contrario la función no será evaluada. El total de requerimientos permiten alcanzar una nota mayor que 7.0, por lo que procura realizar las funcionalidades que consideres necesarias para alcanzar un 7.0. Si realizas todas las funcionalidades y obtienes el puntaje máximo, la nota asignada será igualmente un 7.0. El puntaje de desborde se descarta.**

1. **(0.5 pts TDAs)**. Especificar e implementar abstracciones apropiadas para el problema. Recomendamos leer el enunciado completo (el general y el presentado en este documento) con el fin de que analice el problema y determine el o los TDAs y representaciones apropiadas para la implementación de cada uno. Luego, planifique bien su enfoque de solución de manera que los TDAs y representaciones escogidos sean aplicables y pertinentes para abordar el problema bajo el paradigma funcional.

   Para la implementación debe regirse por la estructura de especificación e implementación de TDA vista en clases: Representación, Constructores, Funciones de Pertenencia, Selectores, Modificadores y Otras Funciones. Procurar hacer un uso adecuado de esta estructura a fin de no afectar la eficiencia de sus funciones. En el resto de las funciones se debe hacer un uso adecuado de la implementación del TDA (ej: usar selectores, modificadores, constructores, según sea el caso. No basta con implementar un TDA y luego NO hacer uso del mismo). **Solo implementar las funciones estrictamente necesarias dentro de esta estructura**.

   A modo de ejemplo, si usa una representación basada en listas para implementar un TDA, procure especificar e implementar predicados específicos para selectores **(ej: en lugar de usar [A | B], realice implementaciones o establezca sinónimos con nombres que resulten apropiados para el TDA. Por ejemplo miSelector([CAR|CDR], CAR) o por ejemplo selectorDeC([A, B, C, D | E], C).**

   Dejar claramente documentado con comentarios en el código aquello que corresponde a la estructura base del TDA. Estructura base mínima que deberá considerar para el resto de los predicados corresponden “system”, “chatbot”, “flow”, “option”, “user” y “chatHistory” que constituyen elementos centrales sobre el que se aplicaran las distintas funcionalidades implementadas.

   Debe contar además con representaciones complementarias para otros elementos que considere relevantes para abordar el problema.

   Especificar representación de manera clara para cada TDA implementado (en el informe y en el código a través de comentarios). Luego implementar constructores y según se requiera, implemente predicados de pertenencia, selectores, modificadores y otros predicados que pueda requerir para las otras funcionalidades listadas a continuación.

   Los predicados especificados e implementados en este apartado son complementarias (de apoyo) a las funciones específicas de los dos TDAs que se señalan a continuación. Su desarrollo puede involucrar otros TDAs y tantos predicados como sean necesarios para abordar los requerimientos.

Procurar que todas las palabras claves creadas se registren en minúsculas o mayúsculas de manera consistente con el objetivo de lograr el match exacto en las interacciones con el usuario. Para tales efectos puede usar funciones como string-downcase, string-upcase o string-ci=? (para comparar case insensitive).

Para cada uno de los ejemplos expresados para las funciones indicadas a continuación, se realizan definiciones con el fin de simplificar las expresiones. Recordar que estas definiciones no son variables, sino que por el contrario se pueden entender como hechos.

2. **(0,2 pts) TDA Option - constructor**. Predicado constructor de una opción para flujo de un chatbot. Cada opción se enlaza a un chatbot y flujo especificados por sus respectivos códigos.

Nombre predicado	option
Prerrequisitos para evaluación (req. funcionales)	1
Requisitos de implementación	Usar estructuras basadas en listas y/o pares. El código de un *option* es único. Su unicidad se verifica al momento de agregarlo a un flow a fin de evitar duplicidad.
Dominio	code (Int) X message (String) X ChatbotCodeLink (Int) X InitialFlowCodeLink (Int) X Keyword (lista de 0 o más palabras claves) X Option
Ejemplo de uso	;creando opciones ;opción 1 vinculada al chatbot 2 con su flujo 4 (asumiendo su existencia) en sistema option(1, “1 - viajar”, 2, 4, [“viajar”, “turistear”, “conocer”], O1). ;opción 2 vinculada al chatbot 4 con su flujo 3 (asumiendo su existencia) en sistema option(2, “2 - estudiar”, 4, 3, [“aprender”, “perfeccionarme”], O2).

3. **(0,5 pts) TDA Flow - constructor**. predicado constructor de un flujo de un chatbot.

Nombre predicado	flow
Prerrequisitos para evaluación (req. funcionales)	2
Requisitos de implementación	Usar estructuras basadas en listas y/o pares Los flujos quedan identificados por un ID único. Su unicidad se verifica al momento de agregarlo a un chatbot a fin de evitar duplicidad. El predicado también verifica que las opciones añadidas no se repitan en base al id de éstos.
Dominio	id (int) X name-msg (String) X Option (Lista de 0 o más opciones) X Flow
Ejemplo de uso	;creando un nuevo flow flow(1, “Flujo 1: mensaje de prueba“, [ ], F1). ;alternativamente podría usarse: flow(2, “flujo 1: mensaje de prueba”, [O1, O2], F2). ; donde O1 y O2 son obtenidos del constructor de *Option*

4. **(0,2 pts) TDA Flow - modificador**. Predicado modificador para añadir opciones a un flujo.

Nombre predicado	flowAddOption
Prerrequisitos para evaluación (req. funcionales)	3
Requisitos de implementación	El predicado también verifica que las opciones añadidas no se repitan en base al id de éstos. Debe retornar false si ya existe.
Dominio	flow X option X flow
Ejemplo de uso	;añadiendo opciones 1 y 2 al flujo 13 ;el resultado alcanzado en F15 es equivalente al ilustrado en F2 del predicado 3. flowAddOption(F13, O1, F14). flowAddOption(F14, O2, F15).

5. **(0,4 pts) TDA chatbot - constructor**. Predicado constructor de un chatbot.

Nombre predicado	chatbot
Prerrequisitos para evaluación (req. funcionales)	4
Requisitos de implementación	Usar estructuras basadas en listas y/o pares Los *chatbots* quedan identificados por un ID único. Su unicidad se verifica al momento de agregarlo a un sistema a fin de evitar duplicidad. El predicado también verifica que los flujos añadidos no se repitan en base al id de éstos.
Dominio	chatbotID (int) X name (String) X welcomeMessage (String) X startFlowId(int) X flows (Lista de 0 o más flujos) X chabot
Ejemplo de uso	;creando un nuevo chatbot chatbot(0, “Asistente”, “Bienvenido\n¿Qué te gustaría hacer?”, 1, FLOWS, CB). ;Donde FLOWS en este ejemplo es la lista de flujos construida con predicados anteriores.

6. **(0,3 pts) TDA chatbot - modificador**. Predicado modificador para añadir flujos a un chatbot.

Nombre predicado	chatbotAddFlow
Prerrequisitos para evaluación (req. funcionales)	5
Requisitos de implementación	Usar recursión de cola o natural para añadir flujos al final de la lista de flujos. El predicado también verifica que los flujos añadidos no se repitan en base al id de éstos.
Dominio	chatbot X flow X chatbot
Ejemplo de uso	;añadiendo flujo a un chatbot ;el resultado alcanzado en CB11 es equivalente al ilustrado en CB11 del predicado 5. chatbotAddFlow(CB10, F12, CB11).

7. **(0,4 pts) TDA system - constructor**. Predicado constructor de un sistema de chatbots. Deja registro de la fecha de creación.

Nombre predicado	system
Prerrequisitos para evaluación (req. funcionales)	6
Requisitos de implementación	Usar estructuras basadas en listas y/o pares El sistema además de contener los distintos chatbots, también contiene el chatHistory de cada usuario que interactúa con el sistema. Sobre el chatHistory, éste corresponde al registro completo del diálogo entre usuario y cada uno de los chatbots con los que interactúa. El historial se mantiene para cada usuario y debe tener el String formateado de cada mensaje del usuario y chatbot (para luego ser visualizado con el predicado write), fecha, hora y emisor (usuario o sistema).
Dominio	name (string) X InitialChatbotCodeLink (Int) X chatbots (Lista de 0 o más chatbots) X system
Recorrido	*system*
Ejemplo de uso	%creando la un nuevo sistema de chatbots con nombre “NewSystem” system(“NewSystem”, 0, [ ], S0). %alternativamente podría usarse: system(“NewSystem”, 1, [CB11], S1).

8. **(0,2 pts) TDA system - modificador**. Predicado modificador para añadir chatbots a un sistema.

Nombre predicado	systemAddChatbot
Prerrequisitos para evaluación (req. funcionales)	7
Requisitos de implementación	Debe verificar que el chatbot no exista en el sistema a partir del id de éste.
Dominio	system X chatbot X system
Ejemplo de uso	%añadiendo un chatbot al sistema. %el resultado alcanzado en s1 es equivalente al ilustrado en s1 de la función 7. systemAddChatbot(S0, CB11, S1).

9. **(0,3 pts) TDA system - modificador**. Predicado modificador para añadir usuarios a un sistema.

Nombre predicado	systemAddUser
Prerrequisitos para evaluación (req. funcionales)	8
Requisitos de implementación	Debe verificar que el usuario no exista en el sistema a partir del id de éste, que está dado por su nombre de usuario (String).
Dominio	system X user (string) X system
Ejemplo de uso	%añadiendo dos usuarios al sistema systemAddUser(S1, “user0”, S2), systemAddUser(S2, “user1”, S3).

10. **(0,3 pts) TDA system**. Predicado que permite iniciar una sesión en el sistema.

Nombre predicado	systemLogin
Prerrequisitos para evaluación (req. funcionales)	9
Requisitos de implementación	Solo pueden iniciar sesión usuarios registrados mediante el predicado de registro de usuarios: *system-add-user*. No se puede iniciar sesión si ya existe una sesión iniciada por otro usuario.
Dominio	system X user (string) X system
Ejemplo de uso	%iniciando una sesión de usuario systemLogin(S3, “user0”, S4).

11. **(0,2 pts) TDA system**. Predicado que permite cerrar una sesión abierta.

Nombre predicado	systemLogout
Prerrequisitos para evaluación (req. funcionales)	10
Requisitos de implementación	Sin requisitos especiales
Dominio	system X system
Ejemplo de uso	%cerrando una sesión de usuario systemLogout(S4, S5).

12. **(1,2 pto) system-talk-rec**. Predicado que permite interactuar con un chatbot.

Nombre predicado	systemTalkRec
Prerrequisitos para evaluación (req. funcionales)	11
Requisitos de implementación	Solo se puede conversar si se ha iniciado una sesión con un usuario previamente registrado.
Dominio	system X message (string) X system ;message puede ser la opción o palabra clave
Ejemplo de uso	%añadiendo un chatbot al sistema. systemTalkRec(S4, “hola”, S5), systemTalkRec(S5, “viajar”, S6),

13. **(0,8 pto) system-synthesis**. Predicado que ofrece una síntesis del chatbot para un usuario particular a partir de *chatHistory* contenido dentro del sistema

Nombre predicado	systemSynthesis
Prerrequisitos para evaluación (req. funcionales)	12
Requisitos de implementación	
Dominio	system X usuario (String) X string *(formateado para poder visualizarlo con write)*
Ejemplo de uso	%solicitando una síntesis del sistema. systemSynthesis(S8, “user0”, Str).

14. **(1 pto) system-simulate**. Permite simular un diálogo entre dos chatbots del sistema.

Nombre predicado	systemSimulate
Prerrequisitos para evaluación (req. funcionales)	13
Requisitos de implementación	El registro de las interacciones queda en chatHistory. La semilla (seed) se establece para generar números aleatorios respetando el principio de transparencia referencial. Usar predicado myRandom provista al final del enunciado. El número máximo de interacciones es una cota superior. Si no se pueden generar más interacciones, la simulación termina.
Dominio	system X maxInteractions (int) X seed (int) X system
Ejemplo de uso	%solicitando una simulación al sistema. systemSimulate(S8, 5, 3212312, S9).

Predicado para generar números pseudoaleatorios:

```
myRandom(Xn, Xn1):-
      MulTemp is 1103515245 * Xn,
      SumTemp is MulTemp + 12345,
      Xn1 is SumTemp mod 2147483648.
```

Script de Pruebas N°1

option(1, "1) Viajar", 2, 1, ["viajar", "turistear", "conocer"], OP1),

option(2, "2) Estudiar", 2, 1, ["estudiar", "aprender", "perfeccionarme"], OP2),

flow(1, "flujo1", [OP1], F10),

flowAddOption(F10, OP2, F11),

% flowAddOption(F10, OP1, F12), %si esto se descomenta, debe dar false, porque es opción con id duplicada.

chatbot(0, "Inicial", "Bienvenido\n¿Qué te gustaría hacer?", 1, [F11], CB0), %solo añade una ocurrencia de F11

%Chatbot1

option(1, "1) New York, USA", 1, 2, ["USA", "Estados Unidos", "New York"], OP3),

option(2, "2) París, Francia", 1, 1, ["Paris", "Eiffel"], OP4),

option(3, "3) Torres del Paine, Chile", 1, 1, ["Chile", "Torres", "Paine", "Torres Paine", "Torres del Paine"], OP5),

option(4, "4) Volver", 0, 1, ["Regresar", "Salir", "Volver"], OP6),

%Opciones segundo flujo Chatbot1

option(1, "1) Central Park", 1, 2, ["Central", "Park", "Central Park"], OP7),

option(2, "2) Museos", 1, 2, ["Museo"], OP8),

option(3, "3) Ningún otro atractivo", 1, 3, ["Museo"], OP9),

option(4, "4) Cambiar destino", 1, 1, ["Cambiar", "Volver", "Salir"], OP10),

option(1, "1) Solo", 1, 3, ["Solo"], OP11),

option(2, "2) En pareja", 1, 3, ["Pareja"], OP12),

option(3, "3) En familia", 1, 3, ["Familia"], OP13),

option(4, "4) Agregar más atractivos", 1, 2, ["Volver", "Atractivos"], OP14),

option(5, "5) En realidad quiero otro destino", 1, 1, ["Cambiar destino"], OP15),

flow(1, "Flujo 1 Chatbot1\n¿Dónde te Gustaría ir?", [OP3, OP4, OP5, OP6], F20),

flow(2, "Flujo 2 Chatbot1\n¿Qué atractivos te gustaría visitar?", [OP7, OP8, OP9, OP10], F21),

flow(3, "Flujo 3 Chatbot1\n¿Vas solo o acompañado?", [OP11, OP12, OP13, OP14, OP15], F22),

chatbot(1, "Agencia Viajes", "Bienvenido\n¿Dónde quieres viajar?", 1, [F20, F21, F22], CB1),

%Chatbot2

option(1, "1) Carrera Técnica", 2, 1, ["Técnica"], OP16),

option(2, "2) Postgrado", 2, 1, ["Doctorado", "Magister", "Postgrado"], OP17),

option(3, "3) Volver", 0, 1, ["Volver", "Salir", "Regresar"], OP18),

flow(1, "Flujo 1 Chatbot2\n¿Qué te gustaría estudiar?", [OP16, OP17, OP18], F30),

chatbot(2, "Orientador Académico", "Bienvenido\n¿Qué te gustaría estudiar?", 1, [F30], CB2),

system("Chatbots Paradigmas", 0, [CB0], S0),

% systemAddChatbot(S0, CB0, S1), %si esto se descomenta, debe dar false, porque es chatbot id duplicado.

```
systemAddChatbot(S0, CB1, S01),
systemAddChatbot(S01, CB2, S02),
systemAddUser(S02, "user1", S2),
systemAddUser(S2, "user2", S3),
% systemAddUser(S3, "user2", S4), %si esto se descomenta, debe dar false, porque es username duplicado
systemAddUser(S3, "user3", S5),
% systemLogin(S5, "user8", S6), %si esto se descomenta, debe dar false ;user8 no existe.
systemLogin(S5, "user1", S7),
% systemLogin(S7, "user2", S8), %si esto se descomenta, debe dar false, ya hay usuario con login
systemLogout(S7, S9),
systemLogin(S9, "user2", S10),
% systemTalkRec(S10, "hola", S11), % si se descomenta, daría false por que “hola” no es un option o keyword
systemTalkRec(S10, "1", S12),
systemTalkRec(S12, "1", S13),
systemTalkRec(S13, "Museo", S14),
systemTalkRec(S14, "1", S15),
systemTalkRec(S15, "3", S16),
systemTalkRec(S16, "5", S17),
systemSynthesis(S17, "user2", Str1),
systemSimulate(S3, 5, 32131, S99).
```

Ejemplo de Output formateador para systemSynthesis(S17, "user2", Str1), write(Str1).

```
1695355052 - user2: hola
1695355052 - Inicial: Flujo Principal Chatbot 1
Bienvenido
¿Qué te gustaría hacer?
1) Viajar
2) Estudiar

1695355052 - user2: 1
1695355052 - Agencia Viajes: Flujo 1 Chatbot1
¿Dónde te Gustaría ir?
```

1) New York, USA

2) París, Francia

3) Torres del Paine, Chile

4) Volver

1695355052 - user2: 1

1695355052 - Agencia Viajes: Flujo 2 Chatbot1

¿Qué atractivos te gustaría visitar?

1) Central Park

2) Museos

3) Ningun otro atractivo

4) Cambiar destino

1695355052 - user2: Museo

1695355052 - Agencia Viajes: Flujo 2 Chatbot1

¿Qué atractivos te gustaría visitar?

1) Central Park

2) Museos

3) Ningun otro atractivo

4) Cambiar destino

1695355052 - user2: 1

1695355052 - Agencia Viajes: Flujo 2 Chatbot1

¿Qué atractivos te gustaría visitar?

1) Central Park

2) Museos

3) Ningun otro atractivo

4) Cambiar destino

1695355052 - user2: 3

1695355052 - Agencia Viajes: Flujo 3 Chatbot1

¿Vas solo o acompañado?

1) Solo

2) En pareja

3) En familia

4) Agregar mas atractivos

5) En realidad quiero otro destino

1695355052 - user2: 5

1695355052 - Agencia Viajes: Flujo 1 Chatbot1

¿Dónde te Gustaría ir?

1) New York, USA

2) París, Francia

3) Torres del Paine, Chile

4) Volver

Ejemplo de Output para:

(display (system-synthesis (system-simulate s0 5 312321) "user312321"))

1695486245 - user312321: Hola

1695486245 - Inicial: Flujo Principal Chatbot 1

Bienvenido

¿Qué te gustaría hacer?

1) Viajar

2) Estudiar

1695486245 - user312321: 1

1695486245 - Agencia Viajes: Flujo 1 Chatbot1

¿Dónde te Gustaría ir?

1) New York, USA

2) París, Francia

3) Torres del Paine, Chile

4) Volver

1695486245 - user312321: 4

1695486245 - Inicial: Flujo Principal Chatbot 1

Bienvenido

¿Qué te gustaría hacer?

1) Viajar

2) Estudiar

1695486245 - user312321: 1

1695486245 - Agencia Viajes: Flujo 1 Chatbot1

¿Dónde te Gustaría ir?

1) New York, USA

2) París, Francia

3) Torres del Paine, Chile

4) Volver

1695486245 - user312321: 1

1695486245 - Agencia Viajes: Flujo 2 Chatbot1

¿Qué atractivos te gustaría visitar?

1) Central Park

2) Museos

3) Ningun otro atractivo

4) Cambiar destino

1695486245 - user312321: 3

1695486245 - Agencia Viajes: Flujo 3 Chatbot1

¿Vas solo o acompañado?

1) Solo

2) En pareja

3) En familia

4) Agregar mas atractivos

5) En realidad quiero otro destino