Fundamentos de Lenguajes de Programación

Versión V:1.0 (Semestre Otoño, 2010)

Objetivos: *Registro referencial.*

1. Introducción

La finalidad de este portafolio es dejar un registro organizado del trabajo docente realizado durante el desarrollo de la asignatura considerada. Este material en su conjunto ejemplifica los distintos instrumentos y herramientas pedagógicas aquí utilizadas.

2. Instrumentos

Se incluyen a lo largo del presente documento los siguientes items:

1. **Prospecto:** Documento que define los lineamientos de esta asignatura.
2. **Normativas:** Se incluyen los distintas normas regulatoria que se consideraron en la asignatura.
3. **Actividades:** Ejemplo de evaluaciones, controles, tareas, pruebas parciales, pruebas recuperativas, pruebas globales.
4. **Gestión Virtual:** Colección de imagenes de la interface digital utilizada en la asignatura.

Fundamentos de Lenguajes de Programación.

Prospecto V:2.0 (Marzo, 2010)

Objetivos: *Los siguientes puntos definen los lineamientos específicos y generales que serán considerados en el desarrollo de este curso.*

1. Sobre este curso

El objetivo de este curso es exponer a los estudiantes que han desarrollado y experimentado con técnicas y conceptos avanzados de programación a algunos principios fundamentales que estan en el centro de cada uno de los lenguajes de programación que han usado. Estos principios yacen en áreas que incluso van má alla de las ciencias de la computación y pueden ser encontrados cada vez que se necesite trabajar y representar algún dominio.

Veremos que un buen lenguaje de programación no es una colección de constructos ad-hoc, y que sin embargo, poseen objetos matemáticos (por ejemplo, expresividad y usabilidad) que son indispensables para establecer las propiedades intrisicas de esto. Veremos como *derivaciones (demostraciones)* y *juicios (afirmaciones)* son un mecanismo universal para hablar y razonar acerca de los constructors propios de un lenguage. Estudiaremos algunos de los principios generales en el diseño de estos, por ejemplo el uso de *tipos* como un principio de organización, demostraciones seguras como noción que nos permite medir la *correctitud* de un programa, así como el estudio diversos mecanismos tales como *funciones*, *registros*, *recursión* y otros más especificos como *polimorfismos*, *excepciones*, *herencias* y *concurrencia*. Estos serán las herramientas que nos permitan diseñar nuestro propio lenguaje de programación.

2. Contenidos

Los siguientes son algunos de los tópicos que estudiaremos durante el semestre:

Juicios, reglas, derivaciones; Definiciones Inductivas, Juicio Hipoteticos; Sintaxis Abstracta y Concreta; Acotamiento y Alcance; Sustituciones, Juicios Generales; Semanticas Estaticas y Dinamicas; Tipeo Seguro; Recursión, Iteración y Puntos Fijos; Tipos de Datos; Tipeo Dinamico; Definibilidad; Polimorfismos; Máquinas Apiladoras, etc

3. Clases

Clase Teórica: Lunes, Miércoles : Módulo 4 [15:40 - 17:10] Sala: 207

Laboratorio: Viernes: Módulo 4 [15:40 - 17:10] Lab:1

4. Equipo de profesores

Profesor Teoría y Laboratorio: Carlos Martínez M. `cmartinezm@ucentral.cl`

5. Recursos

Como materiales de apoyo, utilizaremos el sistema de gestión de cursos Moodle 2008 de la Escuela de Ingeniería en Computación, donde estarán disponibles en detalle, un clase a clase, así como materiales que se irán incluyendo a lo largo del semestre, al mismo tiempo herramientas como foros y anuncios pertinentes de las actividades a desarrollar en este curso.

6. Evaluaciones

El trabajo a realizar durante este curso consiste en Pruebas Parciales, Laboratorios y un Trabajo Final, los cuales serán evaluados con notas en escala de $1,0$ a $7,0$, y cuyos porcentajes están descritos en las siguientes tablas.

$$\begin{aligned}
\text{Pruebas} &= \mathbf{P} & \begin{cases} P_1, & 24\,\% \\ P_2, & 24\,\% \\ P_3, & 32\,\% \end{cases} \\
\text{Quizzes} &= \mathbf{Q} & \begin{cases} C_1, & 4\,\% \\ C_2, & 4\,\% \\ C_3, & 4\,\% \\ C_4, & 4\,\% \\ C_5, & 4\,\% \end{cases} \\
\text{Laboratorio} &= \mathbf{L} & \begin{cases} L_1, & 20\,\% \\ L_2, & 20\,\% \\ L_3, & 20\,\% \\ L_4, & 20\,\% \\ L_5, & 20\,\% \end{cases} \\
\text{Trabajo Final} &= \mathbf{TF}
\end{aligned}$$

El computó de la Nota de Presentación **NP** se determinará a partir de las notas de cátedra **NC** y laboratorio **NL** que acontinuación se definen:

$$\mathbf{NC} = \mathbf{P} + \mathbf{Q} \qquad \mathbf{NL} = \frac{\mathbf{L}\cdot 0{,}3 + \mathbf{TF}\cdot 0{,}2}{0{,}5}$$

desde las cuales;

$$\mathbf{NP} = \begin{cases} \mathbf{NC}\cdot 0{,}6 + \mathbf{NL}\cdot 0{,}4, & \text{Sí } \min\{\mathbf{NC}, \mathbf{NL}\} \geq 3{,}95; \\ \min\{\mathbf{NC}, \mathbf{NL}\}, & \text{De lo contrario.} \end{cases}$$

La Nota Final **NF** será calculada de la siguiente manera, a partir de la nota de presentación **NP** y la Prueba Global **PG**, de la cual se publicará se fecha de realización en forma oportuna.

$$\mathbf{NF} = \mathbf{NP}\cdot 0{,}7 + \mathbf{PG}\cdot 0{,}3$$

Las pruebas consistirán en cuatro problemas de desarrollo. Los laboratorios consistirán en uno o dos ejercicios de desarrollo.

7. Calendario

Las siguientes son las fechas confirmadas de las pruebas parciales, controles, prueba recuperativa y prueba global. Culquier modificación posible se publicará y notificará en forma oportuna.

Pruebas y Controles

Prueba 1	Jueves 15 de Abril
Prueba 2	Jueves 20 de Mayo
Prueba 3	Jueves 24 de Junio

Quizz 1	Martes 20 de Abril
Quizz 2	Martes 4 de Mayo
Quizz 3	Martes 18 de Mayo
Quizz 4	Martes 1 de Junio
Quizz 5	Martes 15 de Junio

Laboratorio 1	Viernes 16 de Abril
Laboratorio 2	Viernes 30 de Abril
Laboratorio 3	Viernes 14 de Mayo
Laboratorio 4	Viernes 28 de Mayo
Laboratorio 5	Viernes 2 de Julio
Trabajo Final	Viernes 9 de Julio

Pruebas Recuperativa y Global

Prueba Recuperativa	Martes 6 de Julio	Prueba Global	Jueves 15 de Julio

8. Referencias

El curso se basará principalmente en los libros aquí referidos, sin embargo la biblografía no es exhaustiva, de considerar una nueva referencia se adjuntará apropriadamente a la información disponible en el sitio MOODLE del curso. Se les recomienda a los alumnos asistir y seguir de cerca las clases y materiales disponibles, y al mismo tiempo estudiar principalmente desde sus propios apuntes y notas de clases.

1. R. Harper. *Programming in Standard ML* (Apuntes), Computer Science Department, Carnegie Mellon University, 2008.
2. R. Harper. *Practical Foundation for Programming Languages* (Apuntes), Computer Science Department, Carnegie Mellon University, 2008.
3. B. Pierce. *Types and Programming Languages.* MIT Press (Primera Edición), 2002.
4. B. Pierce. *Advanced Topics in Types and Programming Languages.* MIT Press (Primera Edición), 2005.

Fundamentos de Lenguajes de Programación.

Normativa Prueba Recuperativa-Reemplazo V:1.0 (Marzo, 2009)

Objetivos: *Los siguientes puntos definen los lineamientos específicos y generales que serán considerados en la prueba reemplazo-recuperativa.*

1. Sobre la Prueba Recuperativa-Reemplazo

Esta prueba tiene por finalidad dar la posibilidad a aquellos alumnos que por alguna razón debidamente justificada requiere obtener una evaluación por aquella prueba no rendida. Al mismo tiempo y en consideración a aquellos alumnos que deseen rendir una nueva prueba, en la que tendrán la posibilidad de reemplazar la evaluación más baja de las tres pruebas rendidas durante el semestre en curso.

2. La evaluación

La prueba consiste de tres partes y cada una de dos problemas, donde cada una de las partes cubre respectivamente el temario de las pruebas rendidas durante el semestre. El alumno deberá en primera instancia elegir y contestar un problema de cada una de estas partes, adicionalmente si el alumno ha faltado **justificadamente** a una de las pruebas, deberá contestar el segundo problema de la parte correspondiente de la prueba faltante.

En resumen, es una *prueba voluntaria* en la medida que el alumno no haya faltado justificadamente a una prueba, y por lo tanto *obligatoria* para todos aquellos que efectivamente justificaron bajo las normas y plazo que la Coordinación de la Escuela de Ingeniería en Computación establece para estos efectos.

3. Un caso hipótetico

Suponiendo que un alumno ha rendido y obtenido las notas siguientes, faltando a la segunda prueba.

Pruebas	P#1	P#2	P#3
Notas	3,0		5,0

- **Caso Justificado:** Notas obtenidas en la *prueba recuperatica-reemplazo*

Preguntas	Notas	Comentario
Pr#1	4,0	
Pr#2		
Pr#3	3,0	P#2: 5,0
Pr#4	7,0	
Pr#5	4,0	
Pr#6		
Nota	**4,5**	

Obteniendo por *recuperación*:

Pruebas	P#1	P#2	P#3
Notas	3,0	5,0	5,0

Finalmente, obteniendo por *reemplazo*:

Pruebas	P#1	P#2	P#3
Notas	4,5	5,0	5,0

- **Caso No-Justificado:**

Preguntas	Notas	Comentario
Pr#1	4,0	
Pr#2		
Pr#3		
Pr#4	7,0	
Pr#5	4,0	
Pr#6		
Nota	**5,0**	

Obteniendo por *reemplazo*:

Pruebas	P#1	P#2	P#3
Notas	3,0	5,0	5,0

Universidad
Central de Chile

Fundamentos de Lenguajes de Programación.

P1

Profesor: Carlos Martínez Méndez

Prueba

26 de Abril, 2010

Nombre: _____

Escribiendo mi nombre adhiero al código de honor.

Pregunta	Puntaje	Nota
#1	$3+3$	
#2	$2+2+2$	
#3	$3+3$	
	NOTA FINAL	

Lea cuidadosamente la siguiente información antes de comenzar el prueba:

- Muestre todo su trabajo claramente y en orden, esto es, si desea obtener el puntaje máximo. Me reservo el derecho de descontar puntos de su respuesta, si no puedo ver como llego a ella (Incluso en el caso en que la respuesta final este correcta).
- **Justifique adecuadamente sus respuestas para asegurar la totalidad del puntaje de la pregunta.**
- Suprima con una raya vertical cualquier página y/o espacio que no ocupe en el desarrollo de sus respuestas.
- Pruebas escritas en lápiz mina no tiene derecho a recorrección.
- La prueba consiste de tres ejercicios, cada uno de ellos es evaluado con una nota de 1 a 7. **Es su responsabilidad confirmar que este folleto contiene el número de páginas adecuado**. Tiene 1:20 minutos para contestar esta prueba, sólo se admiten consultas sobre enunciado.
- Buena suerte!

Declaro no recibir ni entrega información parcial o total que permita la resolución de esta prueba a otra persona realizando esta evaluación: _____

Firma.

1. (*6 points*) (**Funciones sobre** `nat`)

i) Implemente via reglas de inferencia la función exponencial, concretamente mediante el predicado $\texttt{exp}(a; b; c)$ de manera que se cumpla la ecuación matemática esperada, $c = a^b$. Para lograr este objetivo puede asumir la existencia de predicados para la suma y multiplicación: $\texttt{sum}(a; b; c)$, $\texttt{mult}(a; b; c)$ respectivamente.

ii) Considere las siguientes reglas que definen la función $\texttt{fun}(a; b)$ donde $a\,\texttt{nat}$ y $b\,\texttt{nat}$.

$$\frac{}{\texttt{fun}(zero; zero)}\,\texttt{fun}_{\texttt{zero}} \qquad \frac{\texttt{fun}(a; d); \quad \texttt{sum}(\texttt{Succ}(a); d; c)}{\texttt{fun}(\texttt{Succ}(a); c)}\,\texttt{fun}_{\texttt{Succ}}$$

¿Esta reglas implementan correctamente alguna función sobre `nat`? Si es el caso, comente que es lo que calcula dicha función. Justifique su respuesta.

Solución:

(Página adicional)

2. (*6 points*) (Juicios, Reglas, Demostraciones Inductivas)

Considere la definición inductiva de listas con argumentos en los números naturales, `list`:

i) Enuncie las reglas definitorias de la igualdad $a = a\,\texttt{list}$ [Observación: recuerde que los constructos en estas listas son `Nil`, `Cons`]

ii) Desde la reglas en i), enuncie claramente el `principio de inducción estructural` asociado a estas.

iii) Utilice el principio inductivo enunciado en ii) para demostrar la siguiente afirmación referente estas listas:

'Si $\texttt{Cons}(a_1; b_1) = \texttt{Cons}(a_2; b_2)\,\texttt{list}$, entonces $a_1 = a_2\,\texttt{nat}$ y $b_1 = b_2\,\texttt{list}$'

Solución:

(Página adicional)

3. (*6 points*) (Derivabilidad y Admisibilidad)

i) Considere los siguentes juicios hipotéticos, demuestre o refute adecuadamente cada uno de ellos. Aquí asuma que `nat`, `list`, `tree` representa las reglas definitorias, de los números naturales, listas con argumentos números naturales y árboles binarios.

a)

$$\texttt{Succ(zero)}\ \ \texttt{nat} \vdash_{\texttt{nat}} \texttt{Cons(zero, Cons(Succ(zero), Nil))}\ \ \texttt{list}$$

b)

$$\texttt{Succ(zero)}\ \ \texttt{nat} \vdash_{\texttt{nat} \cup \texttt{list}} \texttt{Cons(zero, Cons(Succ(zero), Nil))}\ \ \texttt{list}$$

ii) ¿Qué sucederá con la validez de los juicio hipotéticos correspondiente a *admisibilidad*?

Solución:

(Página adicional)

Universidad
Central de Chile

Fundamentos de Lenguajes de Programación.

L10

Profesor: Carlos Martínez Méndez

Laboratorio

9 de Junio, 2010

Nombre: _____

Escribiendo mi nombre adhiero al código de honor.

Instrucciones

Este laboratorio no tiene evaluación formal, sin embargo, su desarrollo es requisito para obtener la nota final del laboratorio en este curso. La entrega preliminar se hará al final del laboratorio, la versión final se deberá entregar al comienzo del siguiente laboratorio, fecha y hora debidamente conocidos.

Type Safety: Preservación en $\mathcal{L}(\mathtt{num}, \mathtt{str})$

Como hemos visto, una forma de interacción entre el sistemas de tipos y la implementación de evaluación es establecer condiciones que conforman la disposición Type Safety. Esta disposición no es siempre incluida en el diseño de un lenguaje de programación, aquellos que lo considerán se denominan *lenguajes fuertemente tipeados*.

Problema

En la actividad de hoy, desarrollaremos la validación de la primera de las condiciones que conforma Type Safety, la cual se denomina *preservación*

Teorema 1 *Si* $e : \tau$ *y* $e \mapsto e'$*, entonces* $e' : \tau$

[Observación: Se sugiere considerar para la validación mediante inducción estructural sobre las expresiones en $\mathcal{L}(\mathtt{num}, \mathtt{str})$, los siguientes son puntos: Casos bases $\mathtt{num}[n]$, $\mathtt{str}[s]$, Casos inductivos $\mathtt{plus}(e_1, e_2)$, $\mathtt{times}(e_1, e_2)$, $\mathtt{cat}(e_1, e_2)$ y $\mathtt{let}(e_1, x.e_2)$]

Puede utilizar como referencia las notas de R. Harper

Universidad
Central de Chile

Fundamentos de Lenguajes de Programación.

Q4

Profesor: Carlos Martínez Méndez

Quizz

1 de Junio, 2010

Nombre: _____

Escribiendo mi nombre adhiero al código de honor.

Ejercicio: Semántica Dinámica

En el contexto de semánticas dinámicas para el lenguaje $\mathcal{L}(\texttt{num}, \texttt{str})$, tenemos enfoques diferentes de como implementar la función `let`. Las siguientes reglas reflejan estos enfoques:

- Evaluación por *Nombre*:

$$\frac{}{\texttt{let}(e_1; x.e_2) \mapsto [e_1/x]e_2}\ \texttt{let}_n$$

- Evaluación por *Valor*:

$$\frac{e_1 \quad \texttt{val}}{\texttt{let}(e_1; x.e_2) \mapsto [e_1/x]e_2}\ \texttt{let}_{v_1}$$

$$\frac{e_1 \mapsto e_1'}{\texttt{let}(e_1; x.e_2) \mapsto \texttt{let}(e_1'; x.e_2)}\ \texttt{let}_{v_2}$$

Considere las reglas de evaluación establecidas y estudiadas en el capítulo 9. A partir de estas evalue la siguiente expresión bajo cada uno de los enfoques aquí planteados. Comente las diferencias en sus desarrollos, por ejemplo ¿Qué sucede con el número de pasos de sus respectivas evaluaciones?

$$\texttt{let}(\texttt{times}(\texttt{num}[1]; \texttt{num}[2]); x.\texttt{plus}(\texttt{plus}(\texttt{num}[3]; x); \texttt{num}[1]))$$

Solución

Justifique apropiadamente su respuesta. **Tiempo 30 minutos**.

Universidad
Central de Chile

Fundamentos de Lenguajes de Programación.

A2

Profesor: Carlos Martínez Méndez

Actividad

22 de Junio, 2010

Nombre: _____

Escribiendo mi nombre adhiero al código de honor.

Instrucción

En la siguiente actividad tiene por finalidad reforzar los tópicos a evaluar en la tercera prueba parcial.

Ejercicio 1: Semánticas en $\mathcal{L}(\texttt{num}, \texttt{str})$

Considere la siguiente expresión e correspondiente en este lenguaje

$$e := \texttt{let}(\texttt{times}(\texttt{num}[1]; \texttt{num}[2]); \texttt{x}.\texttt{plus}(\texttt{x}; \texttt{times}(\texttt{num}[3]; \texttt{x})))$$

i) ¿Cuál es el tipo de la expresión e?

ii) Evalue según la *semántica estructural*, tanto en su versión *por valor* y *por nombre* esta expresión

iii) Evalue según la *semántica contextual* esta expresión.

Ejercicio 2: Funciones de Orden Superior $\mathcal{L}(\texttt{num}, \texttt{str}, \rightarrow)$

En este ejercicio implementaremos una función de orden superior que nos permita calcular la *composicón* de dos funciones. Considere el siguiente lambda-termino:

$$\mathrm{C} \equiv \lambda \mathrm{f}\mathrm{g}\mathrm{x}.\mathrm{f}(\mathrm{g}(\mathrm{x}))$$

i) Suponiendo que tanto f y g son funciones de $\mathbb{N} \rightarrow \mathbb{N}$. ¿Cuál debería ser el tipo de C?

ii) Determine la implementación de C en nuestro lenguaje.

iii) Considere ejemplos para f y g en la *sintaxis abstracta* y evalue la expresión equivalente a C f g.

Ejercicio 3: Recursión Primitiva $\mathcal{L}(\mathtt{num}, \rightarrow)$

Implemente la función `fact`: $\mathbb{N} \rightarrow \mathbb{N}$ dada por la instancia del esquema de *recursión primitiva*:

$$\boxed{\begin{array}{l}\mathtt{fact}(\mathtt{z}) = \mathtt{z} \\ \mathtt{fact}(\mathtt{S}(\mathtt{n})) = \mathtt{times}(\mathtt{plus}(\mathtt{n}, \mathtt{S}(\mathtt{z})), \mathtt{fact}(\mathtt{n}))\end{array}}$$

i) Encuentre un tipo τ tal que $\mathtt{fact} : \tau$ es valido

ii) Evalue según la *semántica estructural* por valor:

$$\mathtt{fact}(\mathtt{S}(\mathtt{S}(\mathtt{z}))$$

Universidad
Central de Chile

Fundamentos de Lenguajes de Programación

Profesor: Carlos Martínez Méndez

G

Grupos

Junio, 2010 **Nombre:** _____

Grupos de Trabajo.

Tokens
Pablo Rodriguez
Nicolas Díaz

Tema: Generic Programming
Referencia: Cap 18 RH

Parsing
Santiago Morales
Ettiane Salamanca

Tema: Pattern Matching
Referencia: Cap 17 RH

Lambda
Pamela Valdebenito
Felipe Fernandez

Tema: Tipos Producto
Referencia: Cap 15 RH

Juan Pablo Rubilar
Let Claudio Serrano

Tema: Tipos Suma
Referencia: Cap 16 RH

Fundamentos de Lenguajes de Programación.

Clase a Clase V:1.0 (Marzo, 2010)

Objetivos: *Gestión Virtual.*

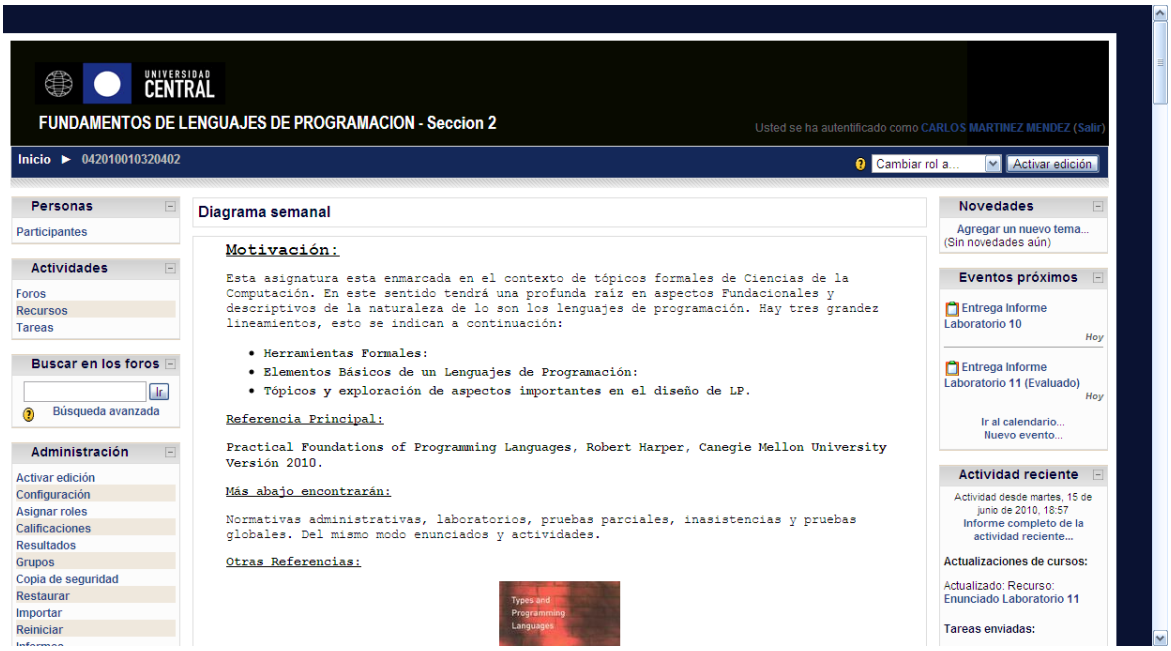
1. Introducción

Los materiales y comunicación a lo largo de la asignatura se establecieron a través de la plataforma de gestión docente denominada MOODLE, la que posee diversos instrumentos pedagogícos tales como generación de actividades online, buzones, mesajería instanánea, etc.

2. Imagenes interface asignatura

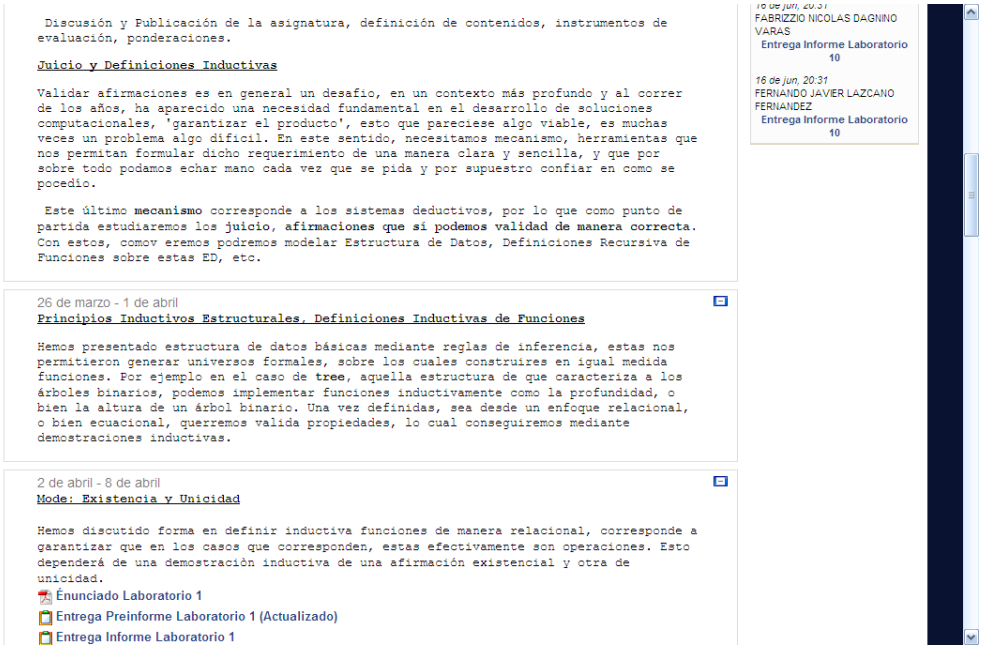
Las siguientes corresponden a imagenes del sitio desarrollado en la plataforma MOODLE y que detallan el plan de trabajo.

1. Introducción y Motivación:

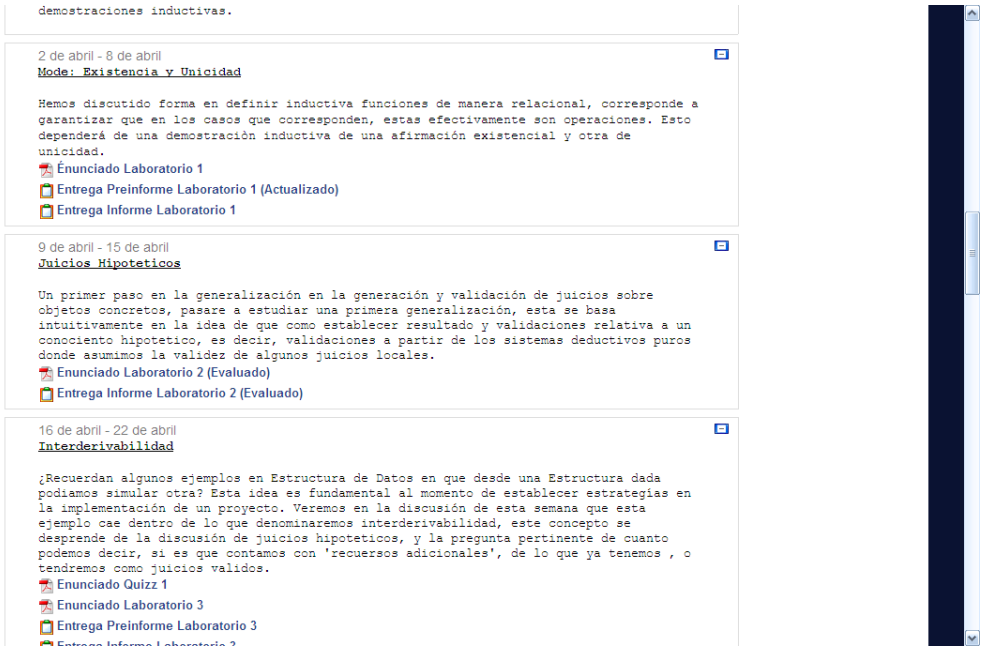


2. Referencias bibliográficas:

3. Preliminares



4. Desarrollo temático


demostraciones inductivas.

2 de abril - 8 de abril
Mode: Existencia y Unicidad

Hemos discutido forma en definir inductiva funciones de manera relacional, corresponde a garantizar que en los casos que corresponden, estas efectivamente son operaciones. Esto dependerá de una demostración inductiva de una afirmación existencial y otra de unicidad.
Énunciado Laboratorio 1
Entrega Preinforme Laboratorio 1 (Actualizado)
Entrega Informe Laboratorio 1

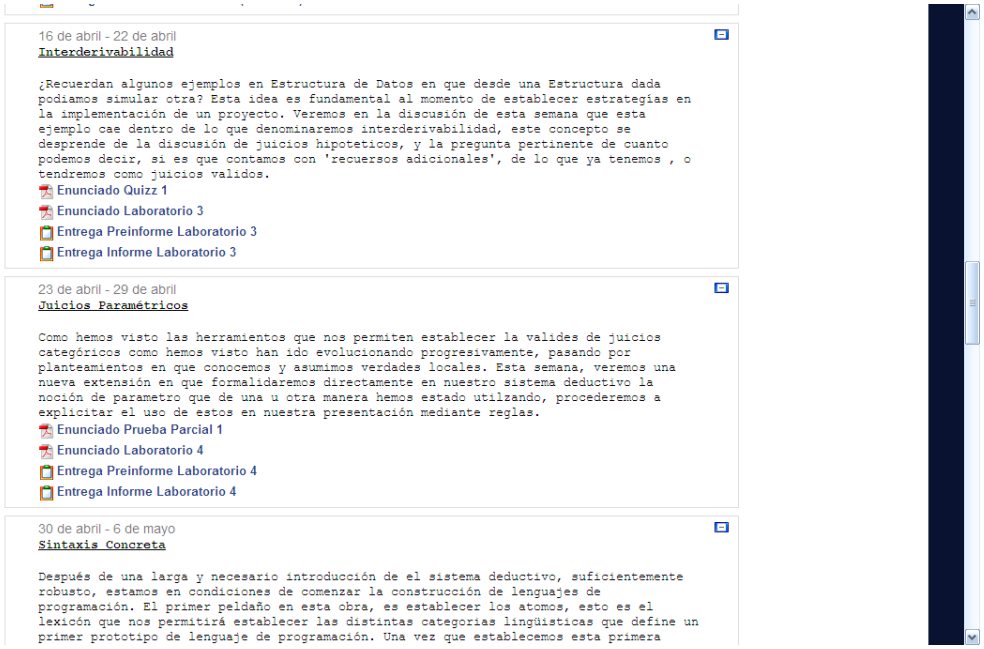
9 de abril - 15 de abril
Juicios Hipoteticos

Un primer paso en la generalización en la generación y validación de juicios sobre objetos concretos, pasare a estudiar una primera generalización, esta se basa intuitivamente en la idea de que como establecer resultado y validaciones relativa a un conociento hipotetico, es decir, validaciones a partir de los sistemas deductivos puros donde asumimos la validez de algunos juicios locales.
Enunciado Laboratorio 2 (Evaluado)
Entrega Informe Laboratorio 2 (Evaluado)

16 de abril - 22 de abril
Interderivabilidad

¿Recuerdan algunos ejemplos en Estructura de Datos en que desde una Estructura dada podiamos simular otra? Esta idea es fundamental al momento de establecer estrategias en la implementación de un proyecto. Veremos en la discusión de esta semana que esta ejemplo cae dentro de lo que denominaremos interderivabilidad, este concepto se desprende de la discusión de juicios hipoteticos, y la pregunta pertinente de cuanto podemos decir, si es que contamos con 'recuersos adicionales', de lo que ya tenemos , o tendremos como juicios validos.
Enunciado Quizz 1
Enunciado Laboratorio 3
Entrega Preinforme Laboratorio 3

5. Disponibildad de Enunciados


16 de abril - 22 de abril
Interderivabilidad

¿Recuerdan algunos ejemplos en Estructura de Datos en que desde una Estructura dada podiamos simular otra? Esta idea es fundamental al momento de establecer estrategias en la implementación de un proyecto. Veremos en la discusión de esta semana que esta ejemplo cae dentro de lo que denominaremos interderivabilidad, este concepto se desprende de la discusión de juicios hipoteticos, y la pregunta pertinente de cuanto podemos decir, si es que contamos con 'recuersos adicionales', de lo que ya tenemos , o tendremos como juicios validos.
Enunciado Quizz 1
Enunciado Laboratorio 3
Entrega Preinforme Laboratorio 3
Entrega Informe Laboratorio 3

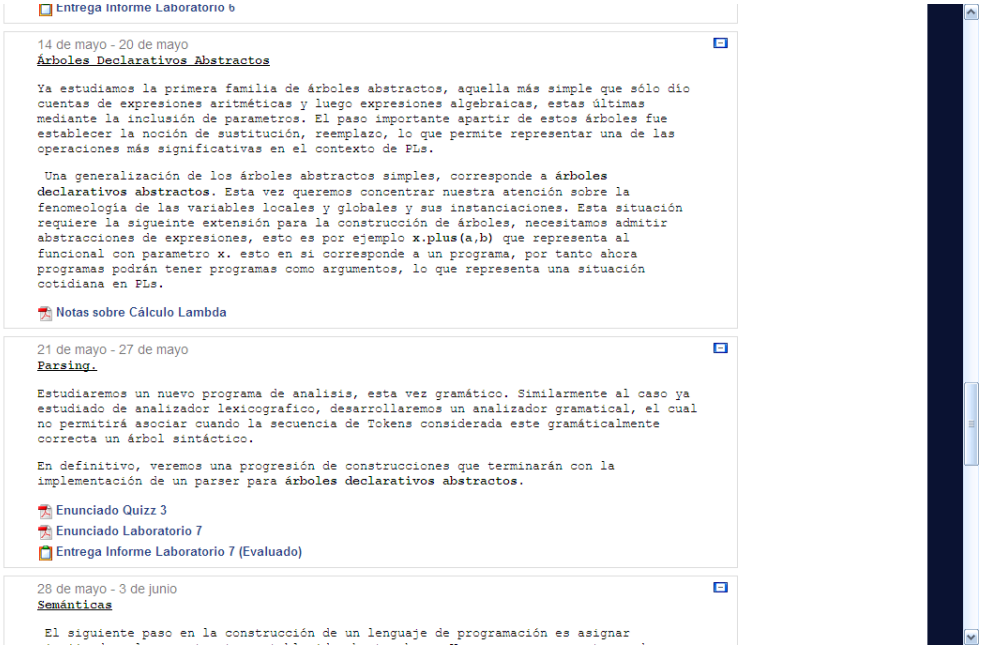
23 de abril - 29 de abril
Juicios Paramétricos

Como hemos visto las herramientos que nos permiten establecer la valides de juicios categóricos como hemos visto han ido evolucionando progresivamente, pasando por planteamientos en que conocemos y asumimos verdades locales. Esta semana, veremos una nueva extensión en que formalidaremos directamente en nuestro sistema deductivo la noción de parametro que de una u otra manera hemos estado utilzando, procederemos a explicitar el uso de estos en nuestra presentación mediante reglas.
Enunciado Prueba Parcial 1
Enunciado Laboratorio 4
Entrega Preinforme Laboratorio 4
Entrega Informe Laboratorio 4

30 de abril - 6 de mayo
Sintaxis Concreta

Después de una larga y necesario introducción de el sistema deductivo, suficientemente robusto, estamos en condiciones de comenzar la construcción de lenguajes de programación. El primer peldaño en esta obra, es establecer los atomos, esto es el lexicón que nos permitirá establecer las distintas categorias lingüisticas que define un primer prototipo de lenguaje de programación. Una vez que establecemos esta primera

6. Actividades dentro del aula y laboratorios



Entrega Informe Laboratorio 6

14 de mayo - 20 de mayo

Árboles Declarativos Abstractos

Ya estudiamos la primera familia de árboles abstractos, aquella más simple que sólo dio cuentas de expresiones aritméticas y luego expresiones algebraicas, estas últimas mediante la inclusión de parametros. El paso importante apartir de estos árboles fue establecer la noción de sustitución, reemplazo, lo que permite representar una de las operaciones más significativas en el contexto de PLs.

Una generalización de los árboles abstractos simples, corresponde a **árboles declarativos abstractos**. Esta vez queremos concentrar nuestra atención sobre la fenomeología de las variables locales y globales y sus instanciaciones. Esta situación requiere la sigueinte extensión para la construcción de árboles, necesitamos admitir abstracciones de expresiones, esto es por ejemplo `x.plus(a,b)` que representa al funcional con parametro `x`. esto en si corresponde a un programa, por tanto ahora programas podrán tener programas como argumentos, lo que representa una situación cotidiana en PLs.

Notas sobre Cálculo Lambda

21 de mayo - 27 de mayo

Parsing.

Estudiaremos un nuevo programa de analisis, esta vez gramático. Similarmente al caso ya estudiado de analizador lexicografico, desarrollaremos un analizador gramatical, el cual no permitirá asociar cuando la secuencia de Tokens considerada este gramáticalmente correcta un árbol sintáctico.

En definitivo, veremos una progresión de construcciones que terminarán con la implementación de un parser para **árboles declarativos abstractos**.

Enunciado Quizz 3

Enunciado Laboratorio 7

Entrega Informe Laboratorio 7 (Evaluado)

28 de mayo - 3 de junio

Semánticas

El siguiente paso en la construcción de un lenguaje de programación es asignar