Trabajo Práctico 2
Programación Orientada a Objetos

Paradigmas de Lenguajes de Programación — 2do cuat. 2022

Fecha de entrega: 8 de noviembre de 2022

1. Introducción

En este taller utilizaremos JavaScript para modelar valuaciones y proposiciones en lógica proposicional. Las valuaciones se representarán como objetos cuyos atributos representan variables proposicionales, y cuyos valores son *true* o *false* seguún el valor que se le asigne a cada variable.

2. Ejercicios

Ejercicio 1

a) Definir la función `esValuacion(val)`, que devuelve *true* si el objeto `val` representa una valuación, y *false* en caso contrario. Tener en cuenta que para que un objeto sea una valuación, todos sus atributos - incluso los heredados - deben tener como valores *true* o *false*.

Ejercicio 2

a) Definir la función `union(val1, val2)` que devuelve una nueva valuación con las variables de ambas valuaciones recibidas y sus valores correspondientes. Si hay variables en común, se considerarán los valores de la primera valuación (es decir `val1`). Las valuaciones originales no deben modificarse.

Ejercicio 3

a) Definir las funciones constructoras `VarProp(nombre)`, `Negacion(prop)` y `OperacionBinaria(operador, f, izq, der)`, que generen objetos que representen variables proposicionales con un nombre dado y operaciones binarias respectivamente.

Las operaciones binarias recibirán un String que indica cómo se escribe su operador, una función de dos parámetros que definirá la forma de evaluar la operación (más detalles en el ejercicio correspondiente), y dos proposiciones que serán los operandos izquierdo y derecho. A modo de ejemplo, ya vienen definidas las funciones `y(izq, der)` y `o(izq, der)`, que crean conjunciones y disyunciones usando la función constructora de operaciones binarias.

Todos los objetos generados con estas funciones - en adelante *proposiciones* - deben poder responder al mensaje `toString()`. La representación de las proposiciones como strings deberá seguir las siguientes reglas:

- La variables proposicionales son representadas por sus nombres.
- Si la representación de una proposición es "P", su negación se escribe como "¬P".
- Las operaciones binarias se escriben entre paréntesis, con su operador en medio de sus dos operandos y separado de ellos por un espacio.
  Por ejemplo, si las representaciones de dos proposiciones son "P" y "Q", su conjunción se escribe como "(P ∧ Q)", y su disyunción como "(P ∨ Q)".

Ejercicio 4

Escribir el código necesario para que las proposiciones puedan responder el mensaje `fv()`, devolviendo el conjunto de los nombres de sus variables proposicionales.

Ejercicio 5

Modificar lo que sea necesario para que todas las proposiciones puedan responder el mensaje `evaluar(val)` para ser evaluadas en una valuación dada. El resultado de la evaluación puede ser `true`, `false` o `undefined` (esto último sucede cuando la valuación no contiene a todas variables de la proposición).

Ejercicio 6

1. Definir la función `implica(izq, der)`, que construya implicaciones entre dos proposiciones. Su operador debe ser "⊃" ("⊃ en HTML").

2. Definir la función `cambiarOperador(prop, operador)` que, dada una proposición, devuelva otra con el mismo comportamiento, pero cuyo operador se escriba con la notación recibida. No se debe modificar la proposición original.

3. Pautas de Entrega

Todo el código producido por ustedes debe estar en el archivo `taller.js` y es el **único** archivo que deben entregar. Para probar el código desarrollado abrir el archivo `TallerJS.html` en un navegador web. Cada función o método (incluso los auxiliares) asociado a los ejercicios debe contar con un conjunto de casos de test que muestren que exhibe la funcionalidad esperada. Para esto se deben modificar las funciones *testEjercicio* en el archivo fuente *taller.js*, agregando todos los tests que consideren necesarios (se incluyen algunos tests de ejemplo). Pueden utilizar la función auxiliar *res.write* para escribir en la salida. Si se le pasa un booleano como segundo argumento, el color de lo que escriban será verde o rojo en base al valor de dicho booleano. Se debe enviar un e-mail a la dirección plp-docentes@dc.uba.ar. Dicho mail debe cumplir con el siguiente formato:

- El título debe ser `[PLP;TP-POO]` seguido inmediatamente del nombre del grupo.

- El código JavaScript debe acompañar el e-mail y lo debe hacer en forma de archivo adjunto con nombre `taller.txt` (pueden cambiarle la extensión para que no sea detectado como posible software malicioso).
- El código entregado **debe** incluir tests que permitan probar las funciones definidas.
- El código de cada ejercicio debe escribirse dentro de la sección correspondiente. No se permite modificar el código de los ejercicios anteriores.

No es necesario entregar un informe sobre el trabajo, alcanza con que el código esté **adecuadamente** comentado (son comentarios adecuados los que ayudan a entender lo que no es evidente o explican decisiones tomadas; no son adecuadas las traducciones al castellano del código). Los objetivos a evaluar son:

- Corrección.
- Declaratividad.
- Prolijidad: evitar repetir código innecesariamente y usar adecuadamente los métodos previamente definidos.

Importante: se admitirá un único envío, sin excepción alguna. Por favor planifiquen el trabajo para llegar a tiempo con la entrega.

Referencias del lenguaje JavaScript

Como principales referencias del lenguaje de programación JavaScript, mencionaremos:

- **W3Schools JavaScript Reference:** disponible online en: https://www.w3schools.com/jsref/default.asp.
- **MDN Web Docs: Mozilla - Referencia de JavaScript**: disponible online en: https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia.