Trabajo obligatorio - Lenguajes de programación

El proyecto consiste en la aplicación de la programación automática, la cual consiste en sintetizar automáticamente un programa a partir de una especificación. Para este trabajo, se trabajará en base a una tabla de verdad dada, para así obtener una expresión de cálculo proposicional.

Estructura del proyecto

El proyecto se encuentra separado en scripts de acuerdo a la fase, o a su utilidad.

Script	Descripción
main.js	Archivo a ejecutar para visualizar programa.
utils.js	Métodos que sirven para un propósito compartido.
prop.js	Estructura de datos a utilizar a lo largo del proyecto.
test.js	Conjunto de pruebas de los métodos de las distintas fases.
fase0.js	Métodos de la fase 0.
fase1.js	Métodos de la fase 1.
fase2.js	Métodos de la fase 2.

Notas del proyecto

- El lenguaje elegido fue JavaScript, en su última versión.
- Fue necesario utilizar algunas bibliotecas, por lo que se recomienda antes de iniciar el programa, ejecutar el comando `npm install`.

Notas de las pruebas

- Las pruebas mostradas en las siguientes secciones se realizaron con una y dos variables.
- Las pruebas se realizan de manera unitaria (por fase) y en conjunto. Ver `test.js`.
- En la sección Extra se encuentra la realización en conjunto de todos los métodos con tres variables.
- Una fase depende de la otra, por lo que al momento de realizar estas pruebas se utilizan las variables definidas con anterioridad para otra fase.

Fase 0

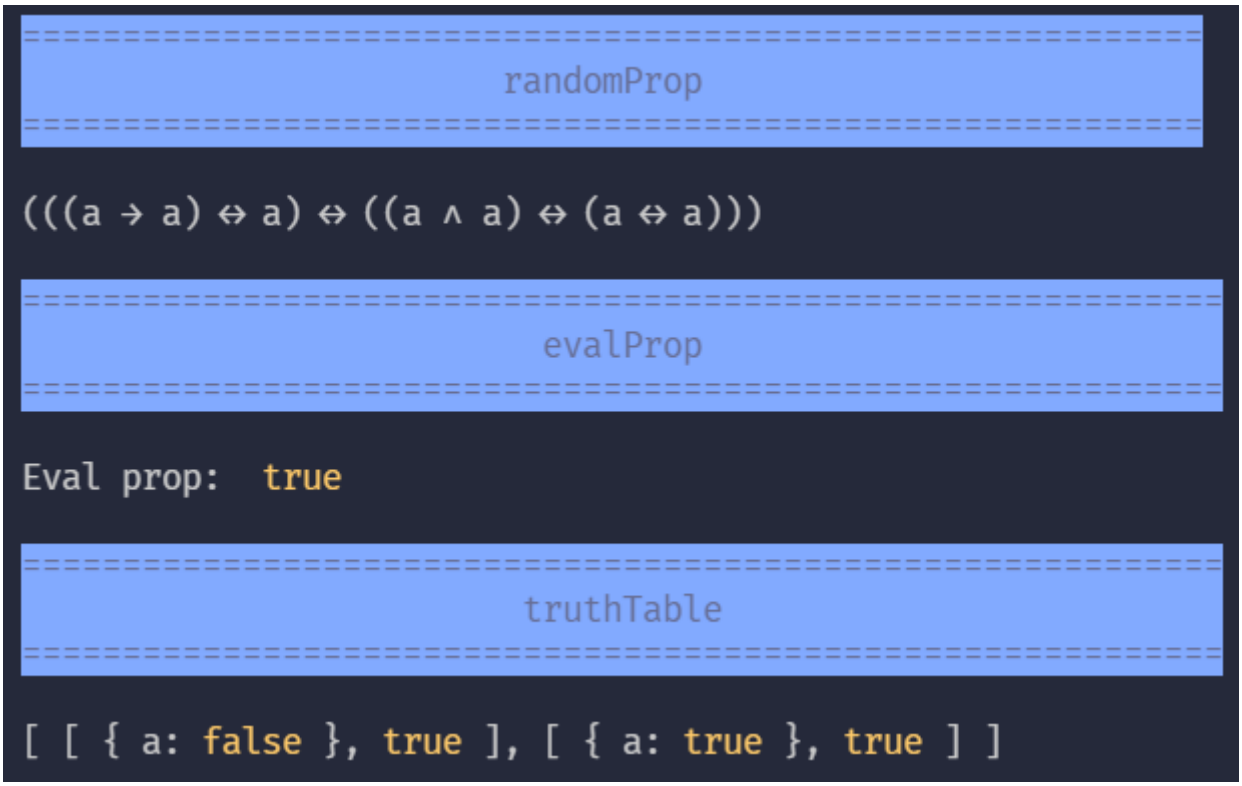
La fase 0 consiste generar una proposición en base a un conjunto de variables definidas, evaluar dicha proposición y generar una tabla de verdad de la proposición.

Para obtener dichas proposiciones fue necesario contar con un generador de números aleatorios, pero en comparación con el comúnmente conocido, este devuelve siempre el mismo conjunto de resultados, lo que se denomina como semilla.

Una variable

```
var rng = prng_alea("LENG.PROG");

var vars = ['a'];
var maxHeight = 4;
var minHeight = 2;

printHeader("randomProp")
let propAleatoria = randomProp(rng, vars, maxHeight, minHeight);
console.log(propAleatoria.toString());

printHeader("evalProp");
console.log("Evaluación de proposición: ", evalProp(propAleatoria, {'a': true}))

printHeader("truthTable");
console.log(truthTable(propAleatoria, vars));
```

Resultado de ejecución con una variable



Dos variables

```
var rng = prng_alea("LENG.PROG");

var vars = ['a', 'b'];
var maxHeight = 4;
var minHeight = 2;

printHeader("randomProp ")
let propAleatoria = randomProp(rng, vars, maxHeight, minHeight);
console.log(propAleatoria.toString());

printHeader("evalProp");
console.log("Eval prop: ", evalProp(propAleatoria, {'a': true, 'b': false}))

printHeader("truthTable");
console.log(truthTable(propAleatoria, vars));
```

Resultado de ejecución con dos variables

```
================================================================
                          randomProp
================================================================

(((a → b) ↔ b) ↔ ((b ∧ b) ↔ (a ↔ b)))

================================================================
                          evalProp
================================================================

Eval prop:  true

================================================================
                          truthTable
================================================================

[
  [ { a: false, b: false }, true ],
  [ { a: false, b: true }, false ],
  [ { a: true, b: false }, true ],
  [ { a: true, b: true }, true ]
]
```

Fase 1

Para esta segunda fase se busca obtener expresiones aleatorias que se asemejen a una tabla de la verdad dada. Esto se realiza mediante la búsqueda aleatoria.

Una variable

```
var vars = ['a'];
var maxHeight = 4;
var minHeight = 2;

printHeader("randomTruthTable & fitness");
let randomTable = randomTruthTable(rng, vars);
console.log("Random truth table: ", randomTable);
console.log("Fitness: ", fitness(arbol, randomTable));

printHeader("randomSearch");
let bestRandomProp = randomSearch(rng, randomTable, 5, {vars, maxHeight,
minHeight});
console.log("Best prop: ", bestRandomProp, fitness(bestRandomProp, randomTable));
```

Resultado de ejecución con una variable

```
================================================================
                    randomTruthTable & fitness
================================================================

Random truth table:  [ [ { a: false }, false ], [ { a: true }, false ] ]
Fitness:  1

================================================================
                          randomSearch
================================================================

Bicondicional {
  left: Bicondicional {
    left: Variable { varName: 'a' },
    right: Negacion { prop: [Variable] }
  },
  right: Condicional {
    left: Condicional { left: [Variable], right: [Variable] },
    right: Conjuncion { left: [Variable], right: [Variable] }
Condicional {
  left: Negacion { prop: Conjuncion { left: [Variable], right: [Variable] } },
  right: Bicondicional {
    left: Condicional { left: [Variable], right: [Variable] },
    right: Disjuncion { left: [Variable], right: [Variable] }
  }
} 0.5
Best prop:  Bicondicional {
  left: Bicondicional {
    left: Variable { varName: 'a' },
    right: Negacion { prop: [Variable] }
  },
  right: Condicional {
    left: Condicional { left: [Variable], right: [Variable] },
    right: Conjuncion { left: [Variable], right: [Variable] }
  }
} 0.5
```

Dos variables

```
var vars = ['a', 'b'];
var maxHeight = 4;
var minHeight = 2;

printHeader("randomTruthTable & fitness");
let randomTable = randomTruthTable(rng, vars);
console.log("Random truth table: ", randomTable);
console.log("Fitness: ", fitness(arbol, randomTable));

printHeader("randomSearch");
let bestRandomProp = randomSearch(rng, randomTable, 5, {vars, maxHeight,
minHeight});
console.log("Best prop: ", bestRandomProp, fitness(bestRandomProp, randomTable));
```

Resultado de ejecución con dos variables

Fase 2