20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by YUVARAJ KUMAR GP
CB.EN.U4CYS21090
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

Outline

LOGICAL PARADIGM

Note on logical programming:
Logic programming is a programming paradigm that is based on formal logic. It allows you to state a program as a set of logical relations. The best-known logic programming language is Prolog.
simple example of a logic program in Prolog:
Facts:
- parent (john, jim).
parent (jim, ann).
parent (ann, bob).
Rule:
grandparent (X, Z) :- parent(X, Y), parent(Y, Z)
In this example, the facts state that John is a parent of Jim, Jim is a parent of Ann, and Ann is a parent of Bob. The rule states that if X is a parent of Y and Y is a parent of Z, then X is a grandparent of Z. You can query the program by asking if a certain relation is true.
For example, you can ask if John is a grandparent of Bob: ?- grandparent(john, bob). true. Prolog will answer true, because John is a grandparent of Bob according to the rules and facts in the program.

Need for Logical Programming

Need for Logical Programming:
1.Declarative Nature
2.Rule-Based Systems
3.Knowledge Representation
4.Constraints and Relationships
5.Natural Language Processing
6.Complex Problem Solving
7.AI Planning and Problem Solving
The Need for Logical Programming:
1.Expert Systems
2.Natural Language Processing (NLP)
3.Databases and Query Languages
4.Constraint Logic Programming
5.Formal Verification
6.Rule-Based Systems

LOGICAL–OZ

Oz is a multiparadigm programming language It supports various programming paradigms, including logic programming, constraint logic programming, object-oriented programming, and concurrent programming.**Oz programming language incorporates logical programming concepts.**

EXAMPLE CODE FOR LOGICAL IN OZ :

```
declare
fun Puzzle X Y Z
X :: 1-9
Y :: 1-9
Z :: 1-9
X + Y = Z
Browse 'Solution: X=' X ' Y=' Y ' Z=' Z
end
Puzzle X Y Z
```

LOGICAL PARADIGM

In this program, we define a puzzle using three variables: X, Y, and Z. We use the :: operator to define the domains of the variables (1 to 9). The constraint $X + Y = Z$ is specified to ensure that the sum of X and Y is equal to Z. Finally, we use the Browse procedure to print out the solution.
X :: 1-9, Y :: 1-9, and Z :: 1-9 specify that X, Y, and Z should be integers between 1 and 9.
$X + Y = Z$ is the constraint that must be satisfied. Browse 'Solution: X=' X ' Y=' Y ' Z=' Z is used to print the solution
To run this program, you would typically use an Oz interpreter or system. Save the code in a file with a **".oz"** extension and execute it using the Oz interpreter. The program will then find and display a solution to the puzzle.

REACTIVE PARADIGM

Note on Reactive Programming: Reactive programming is a programming paradigm that deals with data streams and the propagation of change [0, 1]. It allows you to express static or dynamic data streams and handle real-time updates efficiently. Reactive programming is based on asynchronous programming logic to handle real-time updates to otherwise static content. The main use of the paradigm the value of a variable is automatically updated whenever the values of the variables it depends on change.

simple example in JavaScript to illustrate the concept:

```
let b = 1;
let c = 2;
let a = b + c;
console.log(a); // 3
b = 10; console.log(a); // Still 3, as 'a' is not a reactive variable in this case
```

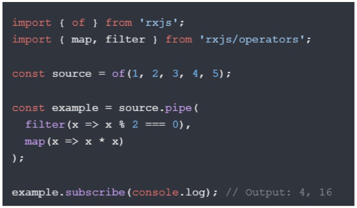
REACTIVE PARADIGM

Now, let's imagine a special operator that changes the value of a variable not only when explicitly initialized but also when referenced variables are change
let b = 1;
let c = 2;
let a = b + c;
dollar symbol need to come because it the operater which help in the console parre
console.log(a); // 3
then the code will look likes
let a =@ b+c instread of @ dollar symbole should come
b = 10;
console.log(a); // 12, as 'a' is now a reactive variable

REACTIVE

what is Reactive Programming : Reactive programming is a declarative programming paradigm that is based on the idea of asynchronous event processing and data streams.The main use of the paradigm the value of a variable is automatically updated whenever the values of the variables it depends on change

simple example of reactive programming using RxJS, a popular reactive programming library:



```
import { of } from 'rxjs';
import { map, filter } from 'rxjs/operators';

const source = of(1, 2, 3, 4, 5);

const example = source.pipe(
  filter(x => x % 2 === 0),
  map(x => x * x)
);

example.subscribe(console.log); // Output: 4, 16
```

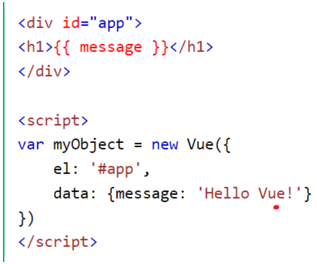
In this example, source is an observable sequence of numbers. The filter operator is used to filter out odd numbers, and the map operator is used to square the remaining numbers. The subscribe method is then used to print the resulting numbers.

The Need for Reactive Programming and real time usage

The Need for Reactive Programming:
1.Asynchronous Operations
2.Event Handling
3.Real-time Applications
4.Responsive User Interfaces
5.Error Handling
6.Concurrency and Parallelism
Real Time usage of reactive:
1.Financial Trading Systems
2.IoT (Internet of Things) Applications
3.Live Sports Updates
4.Collaborative Editing and Document Sharing
5.Online Chat and Messaging Applications

vue.js

what is vue.js :
Vue.js is a JavaScript framework used for building user interfaces and single-page applications. It is designed to be incrementally adaptable, which means you can use as much or as little of it as needed, and it is easy to integrate into other projects. Vue.js is often praised for its simplicity, flexibility, and a gentle learning curve



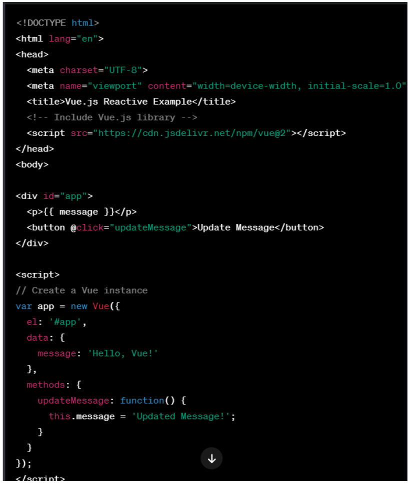
```
<div id="app">
<h1>{{ message }}</h1>
</div>

<script>
var myObject = new Vue({
    el: '#app',
    data: {message: 'Hello Vue!'}
})
</script>
```

vue.js

1.In the example below, a new Vue object is created with new Vue().
2.The property el: binds the new Vue object to the HTML element with id="app".
REACTIVE in vue.js
In the context of Vue.js, reactive programming refers to the framework's ability to automatically track changes in the data and efficiently update the user interface (UI) in response to those changes. Vue.js achieves reactivity through its reactivity system, which is one of the core features of the framework.

REACTIVE in vue.js



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0"
  <title>Vue.js Reactive Example</title>
  <!-- Include Vue.js library -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
</head>
<body>

<div id="app">
  <p>{{ message }}</p>
  <button @click="updateMessage">Update Message</button>
</div>

<script>
// Create a Vue instance
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello, Vue!'
  },
  methods: {
    updateMessage: function() {
      this.message = 'Updated Message!';
    }
  }
});
</script>
```

Figure: reactive in vue.js

REACTIVE in vue.js

In this example:
1.We include the Vue.js library using a CDN.
2.We create a div with the id "app," which will be the root of our Vue application.
3.Inside the div, we have a paragraph (<p>) that displays the message property and a button (<button>) with an @click directive to call the updateMessage method.
Here's how reactivity works in this example:
1.The data option in the Vue instance contains the reactive data. In this case, there is a message property initialized with the string 'Hello, Vue!'.
2.The message expression in the paragraph binds to the message property. This creates a reactive dependency.
3.Clicking the "Update Message" button triggers the updateMessage method, which modifies the message property.
4.Vue.js automatically detects the change in the message property and updates the corresponding part of the user interface, reflecting the new value.

Logical vs Reactive Programming

Logical Programming	**Reactive Programming**
Follows a declarative paradigm without explicitly defining the flow of control.	Follows an event-driven paradigm reacting to changes and events.
Relies on a logical inference engine for solutions based on facts and rules.	Involves handling asynchronous data streams and real-time reactions.
Emphasizes logical expressions, facts, and rules for relationships and constraints.	Involves expressing reactions to events, often using observers.
More declarative, focusing on what needs to be achieved.	Can involve a mix of declarative and imperative styles.
Commonly used in AI, expert systems, and knowledge representation.	Applied in user interfaces, real-time systems, and event-driven scenarios.

Difference Between Oz and Vue.js

Oz	Vue.js
Multiparadigm language for concurrent and distributed programming.	JavaScript framework for building user interfaces.
Supports logic programming, constraint programming, and object-oriented programming.	Focused on declarative rendering and reactive UI for web applications.
Emphasizes high-level abstractions for expressing complex relationships.	Component-based architecture with a focus on simplicity and reactivity.
Specifically designed for concurrent and distributed programming.	Primarily used for client-side applications; additional tools may be needed for server-side concurrency.
Used for distributed systems, expert systems, and logic-based applications.	Commonly used for building modern web applications, especially single-page applications.
May have a steeper learning curve, especially for logic programming.	Known for a gentle learning curve, suitable for developers familiar with HTML, CSS, and JavaScript.

Similarities: Logical vs Reactive Programming

Aspect	Description
Declarative Style	Emphasizes specifying what needs to be achieved rather than explicit step-by-step procedures.
Event-Driven Nature	Inherently responds to changes and events in the system.
Asynchronous Operations	Frequently involves handling asynchronous operations and concurrent execution.
Pattern Matching	Common feature used for matching rules against facts or handling data streams.
Functional Programming Concepts	May incorporate functional programming concepts like higher-order functions.
Concurrency and Parallelism	Can involve aspects of concurrency and parallelism in handling multiple events concurrently.
Data Transformation	Involves transforming and manipulating data based on logical relationships.
Statelessness	Often promotes a certain level of statelessness, emphasizing immutable data.

Differences: Logical vs Reactive Programming

Aspect	Description
Purpose and Domain	Logical programming (Oz) designed for concurrent and distributed programming. Reactive programming (Vue.js) focuses on user interfaces.
Programming Paradigm	Oz supports logic, constraint, and object-oriented programming. Vue.js focuses on declarative rendering for UI.
Concurrency and Distribution	Oz designed for concurrent and distributed programming. Vue.js primarily for client-side applications.
Application Type	Oz used for distributed systems, expert systems, etc. Vue.js for modern web applications.
Learning Curve	Oz may have a steeper learning curve. Vue.js known for simplicity and a gentle learning curve.

CONCLUSION

logical programming in Oz and reactive programming in Vue.js serve different purposes and are applied in different contexts. Logical programming is often associated with **rule-based systems and artificial intelligence**, while reactive programming is commonly used for **building dynamic and interactive user interfaces**. Developers choosing between these paradigms should consider the specific requirements and characteristics of their projects.

Bibliography

REFERENCES:

1.https://www.geeksforgeeks.org/difference-between-functional-and-logical-programming
2.https://softwareengineering.stackexchange.com/questions/216554/how-is-reactive-logic-programming-different-from-functional-programming
3.https://www.doc.ic.ac.uk/ cclw05/topics1/summary.html
4.https://www.linode.com/docs/guides/logic-programming-languages
5.https://www.computerhope.com/jargon/l/logic-programming.htm
6.https://www.techtarget.com/searchapparchitecture/definition/reactive-programming
7.http://mozart2.org/mozart-v1/doc-.4.0/tutorial/index.html
8.http://mozart2.org/
9.https://www.ps.uni-saarland.de/oz2/

PROMPT:

1.logicalprogrammingexamples
2.simpleexampleof logicalprogramming
3.Reactiveprogramming
4.ozisakernallanguage
5.ozlanguageexamplecode
6.examplecodef orvue.js