INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Esterel Is An Imperative, Synchronous And Reactive Programming Language

Samir Kumar Rout
M.Tech scholar,
NM Inistitute of Engineering and Technology, BPUT,Odisa,India

Rajeeb Kumar Jena
M.Tech scholar,
NM Inistitute of Engineering and Technology, BPUT,Odisa,India

Abstract:

Many embedded systems belong to the class of reactive systems, which actively reacts to the environmental inputs and generates correct and effective outputs. However in this paper we will keep our discussion limited to a known Synchronous & Reactive language Esterel. Esterel is both a programming language, dedicated to programming synchronous, reactive systems, and a compiler which translates Esterel programs into finite-state machines. It belongs to the category of Synchronous languages, like SyncCharts, Lustre, Argos or Signal. These languages are particularly well-known to be used for programming reactive systems. Which also includes real-time systems and control automata

This paper aim is to make an idea that when a language (Esterel) already reviewed as Imperative language that means it works like Imperative style and also reviewed as Synchronous & Reactive language individually then why not it is called as the combination of both Imperative, Synchronous & Reactive language. For that we have to just follow the mechanism of different reference papers and composite different angle of the language (Esterel) that it is imperative language, it is Synchronous & Reactive language. Then we had just made coordination that 'Esterel is an Imperative, Synchronous and Reactive Language.

1.Introduction

A real time system is one whose correctness involves both the logical correctness of outputs and their timeliness. An embedded system is some combination of computer hardware and software either fixed in capability or programmable for ex: camera, cellular phones, key-board, mouse etc. For designing a real time embedded system it will need real time operating system inside the system as a system program which enables to interact with its environment synchronously. There are different types of synchronous languages such as: Argos, Atom, Averest, Chuck, Esterel, Labview, LEA, Lustre, PLEXIL, SINGAL, SOL etc. But the first synchronous programming languages were Esterel, Lustre and Singal which are invented 80s in France.

Plentiful research paper prove that Esterel is a reactive programming language but my review is that Esterel is not only a reactive programming language but also it is a Imperative, Synchronous and reactive programming language for that we had take three parameter and each parameter individually prove its emergencies.

2. Imperative Programming

Imperative Programming language defines sequences of command/instruction for the computer to perform. It describes computation in terms of statement that change a program state (memory state).The commands is similar to the machine instructions or it is closer to the machine representation. It is a model that is based on moving bits around and changes the machine state [6]. Esterel adopts a classical imperative style [4] & it is a mode that is based on moving bits around and changing machine state. It handles classical assignable variables that are local to concurrent statement and cannot be shared (signals that are used to communicate with the environment and between concurrent process).signal carries a status which shows its presence or absence in a given reaction which is determined by the value carried by the signal.

As we know a Imperative Programming language must be consisting of two things first one in an program state that we have already discuss in above and second one is an instruction that change the program state (memory state)[5].an instruction need some parameters such as variables ,operators, control flow etc .So Esterel provides standard imperative statements such as assignments, signal emission, sequencing, conditional loop, trap-exit(exception-block).these statements are executed on an infinitely fast machine, and temporal statements such as triggers (await event do.....) watchdogs(do....watchdogs events) or temporal loop(loop......each event).Esterel consists of freely mixing independent time scales. For an example-watchdog which is defined how long their body will be executed.

Now verify how Esterel statement change program state (memory state) for that we have to need that the language supports integers, basic arithmetic operations, assignment, sequencing, looping and branches etc. Esterel has provided 3 types of primitives such as integer, Boolean (with constants true or false) and triv (with a unique constant also called triv) .the user can declare his own abstract types by writing them after the type keyword.

- type double, time;
- type integer, age; etc.

Here is an example of Esterel programming style with description:

```
   Module XYRO;
           Input X, Y, R;
           Output O;
            Loop
                Abort
                 (Await    X    ll
await Y)
                      Emit O
              When R
    End.
```

Figure 1.1

In the above program there are number of things: first there is an explicit “parallel” (II)programming construct; then there are specific signal, and signals can be emitted o received to progress control, beyond some points. The main paradigm beyond the synchronous language philosophy is that time is divided into discrete (logical instants), and these instants are shared equally between components and that the joint behaviours created during an instant is called a reaction. The issue of simultaneity between several events (signal occurring during the same reaction) and proper handling of priorities and causality between such simultaneous events is fully considered in the semantics. Here the output O will be emitted each time signal X and Y have been notified simultaneously or one after the other from the environment. Basically this is happen unless the reset signal

R occurs which erases all effects from previous receptions of X and Y, including the ones at the current instant.

The instruction await signal can further be expanded as abort loop pause when signal. This explores the specific pause statement. Which is the signal primitive statement that allows dividing the progress of executions over time into successive instants Then any emitted signal be it sensed from the external environment or explicitly emitted from a parallel branch inside is considered as present for a lifetime of that instant (only). Signal presence is broadcast thus perceived uniformly all across the program, parallel subprograms may proceed with their activities while put under the ground of a possible interrupting signal (as in about p when s). The fine turning and precise handling of priorities is allowed by the primitive constructs (as in week abort p when s vs abort p when s) [1].

The first pre-empting its body only at the end of the reaction while the next discards it right away for the reaction when S occurs.

Various primitive constructs allow starting; freezes abort or reset a subprogram as needed by the problem specified. Esterel is based on a clean notion of instantaneous causality, which allows discard some program as ill-formed. The prototype ill constructs are present S then emit S and present S else emit S.where forward executions are needed to validate or invalidate a branching decision based on signal.

Data handling in Esterel is defined to C/C++ as host language. Even though the presentation was historically drastically this is similar to system ‘C’ where several primitive constructs and synchronous tasks communicating as to form the global program/ system behaviour.

In the basic of above description it is to be proven that ESTEREL is an Imperative Programming language because ESTEREL has power to change the machine state like any assemble language. It will directly work with the memory block of the system in standard imperative style.

3. Synchronous programming

Synchronous programming languages were invented in France in the 80s.Esterel, lustre and signal. A synchronous program reacts to its environment in a sequence of ticks and computations within a tick are assumed to be instantaneous. For an example “every 60 second emit minute” specifies that the signal “minute” is exactly synchronous with the 60th occurrence of the signal “second”. It is responsible for formal analysis, verification and auto code generation and it is done by the timer that synchronises on a fixed period basis. There should be some waiting time before re-synchronisation. More over a logical instants can be seen as ‘time stamps’ that indicate the date/time/period at which events occurs. Since they are ordered (at least partially) which respect to a given reference set of instants. The date/time/period comparison becomes possible. On the basis of the observed occurrences of events one can also determine very easily the frequency at which events occurs during an execution of the system. This information is captured by the notion of logical or abstract clock which consists of a set of logical instants obtained from the system vision. It serves to synchronize the occurrence rates of expected events in the system.

The main design and programming issues of the synchronous language are [3]:

- Mathematical Specification: Such specifications can be confidently considered to reason a priority about the system behaviours.
- Determinism And Predictability: These respectively concern the specification of functional aspects and of timing aspects of a system.
- Concurrency And Hierarchy: An embedded system often involves concurrently with regard to its environment. It is commonly seen as being formed of several components running in parallel to achieve the task/goal which has been assigned to the system.
- Platform-Independent Designs: Synchronous language offer platform-independent descriptions which are strongly retarget able to different implementation platforms.
- Automatic Code Generation: To avoid error-prone manual programming and tedious debugging tasks, the possibility to automatically generate code from high-level and proven correct specifications is a very interesting solution. All synchronous languages offer this facility.
- Uniform Design Frameworks: The synchronous languages provide the designer with a way to describe at high level using the same formation of the functions on algorithms achieved by the systems, and a model of the h/w infrastructure to be considered for implementation.

The implementation models for system languages are event-driven and clock driven execution [9]. In event driven each reaction is initiated on the occurrence of some input event. This may consist in seeing the watchdog as only activated wherever it receives a request event req(a request signal from environment)from the nuclear power plant(environment) then ‘compute reaction’ amount to executing some statements that typically modify the variables manipulated in the program and to computing the output data depending on the current state of input data and finally the next state of the program is updated via its associated memory information .here the memory information is first initialized before it enters the main loop. But in clock driven reactions are only initiated by abstract clock ticks. For instance this may consist in seeing the watchdog as only activated whenever it receives an input tick event from the external/global clock. It notice that abstract/global clock ticks can be equivalently represented as pure input events meaning events which only indicate that the system must react. But which do not carry all functional information required to compute the system outputs. Moreover both event-driven and clock driven implementation models assume that all actions considered take bounded memory and time capacities. Generally synchronous languages are associated with powerful compiles that enable automatic code generation from high level specifications.

Now observe partial design as can be conceived with the SINGAL compiler.

```
initialize memory;
for each input event do
        compute reaction;
         update memory;
end;
```

Figure 1.2:Event-driven

```
initialize memory;
for each clock tick do
        read inputs;
        compute
reaction;
         update
memory;
end;
```

Figure 1.3: Clock- driven

Now we have to check how ESTEREL fulfil all the design & programming issues of a synchronous language for that we have to take an example which describes machine specification of a process. In the program which shows in fig-1.4 represents machine state like.

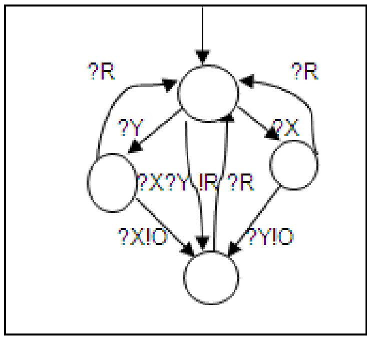

Figure 1.4: Machine Specification Done By A Finite State Machine

According to the machine state this program emits the output signal ‘O’ when the input signals ‘X’ and ‘Y’ have been received in any order that may be simultaneously. Whenever the input signal ‘R’ (reset signal) is received the behaviour of the program is immediately reinitialized. At each time when signal ‘R’ is received it will reset the signal which indicate assignment of signal that is satisfy the property of mathematical specification and also the timing aspect of the system as the program is executed simultaneously when both signal ‘X’ & ‘Y’ are received it is indicates the parallel computing objective of the synchronous programming. for automatic code generation ESTEREL source program can be compiled into either automata [12], electronic circuits [13], or an intermediate format in the form of a control flow graph [14] because all the compiler particularly consist to analysis set of abstract clock which constraints for allows one to verify typical behavioural properties such as the general part of the compilation during which program can be synthesized in formats and it is fever automatic code generation.

In the basic of above description it is to be proved that ESTEREL is an synchronous language because the execution style of signal (data) will be done in a sequence of time interval that may be varies signal to signal according to the environment .but it will satisfy most of the design & programming issues that is define by a synchronous language so we have to told that ESTEREL is a synchronous language.

4. The term ‘reactive system’

The term ‘reactive system’ was introduced by David Hanel and Amis Prueli [1].reactive system means it have to react to its environment which cannot wait. Many computer applications involve programs that maintain a permanent interaction with their environment, execution (reaction) to inputs coming from this environment by sending output to it. When clock ticks, a reaction occurs which compute the output signal and the new state of the program from the input signals from the current state of the program [7]. It may finish the execution instantly or delay part of it till the next instant, because it reached at least one pause(stops the execution till next instant)instruction in the latter case the execution is resumed when the clock ticks again from the locations of the pause instructions reached in the previous instant[2]. A system is called as reactive system whose main component is a reactive program such as: Real time process controller units, signal processing units, Morden digital watches, videogames. Operating system drives mouse/key-board interface driver communication protocol emitters and receivers are also examples of reactive programs (embedded in complex system) [8].

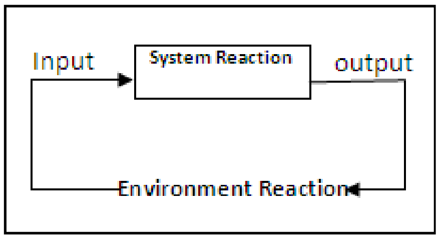

Figure 1.5

In the above fig-1.5 shows that how a reactive system behaves in its environments & it is representing three things which complete a reactive system such as [10]:

A: An input & output interface: which enable the program to interact with environment for input reception and output production it handles interrupts, reads sensors, activates effectors and transforms external physical events into internal logical ones and vice -versa.

B: A reactive kernel (program): which contain the logic of system? It controls the logical inputs and outputs in deciding what computations and what outputs must be generated in reacting to input.

C: A data handling layer: Performs classical computations requested by the reactive kernel.

A reactive program indicates collection of execution, each execution is sequence of reactions and each reaction performs on the memory in form of memory state according to the input/output (Boolean, integer or floating values).

An ESTEREL statement runs in steps called reactions in response to the ticks of a global clock and each reaction takes one instant. Before we discuss about how ESTEREL implemented reactive system we have to know that how a reactive system commonly implemented for implementation of reactive system there are 2 ways such as event driven & clock driven. Take a sample of implementation of event driven in ESTEREL.

```
<initialize memory>
(for each input event do)
        <compute output>
         <update memory>
end;
```

Figure 1.6

```
Every 60 SECOND do
   Emit MINUTE
END
```

Figure 1.7

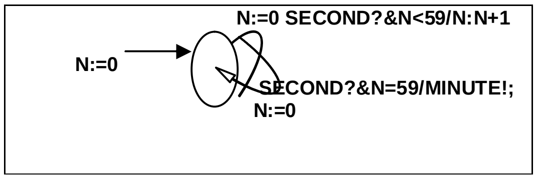

Figure 1.8.

Here in fig-1.6 shows the ESTEREL program of fig-1.7 both are same in form of sequence & reaction like the detail shows in fig-1.8 that first the ‘N’(new instant of time) is initialized to complete the procedures till the value of ‘N’ is not less than 60 once the value of ‘N’ reach above the 59 then it will execute(emit) the *minute* signal then reaction occur at each instant of initialize action of value(Boolean, integer etc)so it is indicates that the style of programming in ESTEREL is reactive . There is no of statement which helps to programming like reactive such as ‘do........watching’. The ‘do.......watching’ statement kills the body of the program as soon as the signal looked for becomes present.ie:

```
do
  awit        BUTTON
  emit        ACTION
  watching SECOND
```

Figure 1.9

Here if SECOND(signal) is present before BUTTON(signal) or at the same instant, then Action(predefine reaction of the system) is not emitted and the watching terminates, suppose the BUTTON(signal)is present before SECOND(signal), then ACTION is immediately emitted and the watching terminates, otherwise if neither SECOND(signal) nor BUTTON(signal) are present then nothing is done and the waiting for BUTTON(signal) and SECOND(signal) is postponed to the next instant.

ESTEREL program mainly composisation of four things such as[2]:

- Reactive
- Automicity of Reactions.
- Instantaneous broadcast.
- Determinism.

But we have to justify only two things Reactive & Atomicity of reactions which enable our paper aim that ESTEREL is a reactive programming language.

4.1.Reactivity

The basic model of ESTEREL is reactive model which indicates a communicating system that continuously interacts with its environment. When there is an input event activated then the reactive system reacts (action perform) according to output events. Reactive system are such as collection of pre-define reactions (react) in related to the input events so that the input events are inputted (receive) by the input lines and the output events produce by the output lines. A reactive system works in basis of reactive statements. ESTEREL reactive statement such as: ‘await E’ it will indicates that stops the execution until the first instant signal ‘E’ becomes present. The most basic ESTEREL reactive statement is the watching statement which shows that a generalized watchdog.

4.2.Atomicity Of Reactions

Atomicity of reactions indicates the basic clock of activations or in other words it allows to consider reactive programs as pre-define schedule a reactive program is a collection of reactive and non-reactive statements that enable to produce a sequence of output events from a sequence of input events. for an example: if input events are:’i1, i2, i3..........in’ are inputted then ‘o1, o2, o3 ...on’ are output events which are produce according to the specification.

In the basic of above description it is to be proved that ESTEREL is an reactive language because it had contain the attribute & behaviours of an reactive system along with fulfil the hypothesis objective of an reactive system so ESTEREL is an reactive system.

5. Conclusion

A simple conclusion is that when a language is enable to interact with the machine (computer) in an assemble language style which is known as imperative style & secondly it should transfer/permitted the inputs/process in a synchronous manner then it is known as process (execution) synchronisation. Then finally the result/final product will be produced in a fixed/predefine/determine way that is why it is known as reactive.

Each parameter of the paper proved it that; Esterel interacts with machine (computer) in an Imperative style & not only in synchronous manner but also in a reactive way. So our conclusion is That Esterel is an Imperative, synchronous and reactive programming language.

6.Reference

1. Esterel, Robert de Simone, INRIA Sophia-Antipolis. http://www.esterel-eda.com/
2. The ESTEREL Language by FREDERIC BOUSSINOT and SIMONE, PROCEEDING OF THE IEEE, VOL, 79, NO.9, SEPTEMBER 1991.
3. Synchronous Programming: Overview, A. Gamatié, Designing Embedded Systems with the SIGNAL Programming Language: Synchronous, Reactive Specification, DOI 10.1007/978-1-4419-0941-1_2, _c Springer Science+Business Media, LLC 2010
4. The ESTEREL synchronous programming language: design, semantics, implementation, Gkard Berry and Georges Gonthier, Ecole des Mines and INRIA, Sophia-Antipolis, 06565 Valbonne, France
5. Imperative Programming, (Class note by IIT).
6. The Esterel v5_92 Compiler, http://www-sop.inria.fr/esterel.org/.
7. A Deterministic Logical Semantics for Esterel, Olivier Tardieu, URL-www.elsevier.nl/locate/entcs.
8. Synchronous programming, www.didel.com/doc/DopiSync.pdf.
9. Synchronous Programming of Reactive Systems*, A Tutorial and Commented Bibliography by Nicolas Halbwachs.
10. Synchronous Programming Of Reactive System by Pascal Raymond, Nicolas Halbwachs, Verimag-CNRS.
11. Esterel Studio Update by Kim Sunesen,www.esterel-eda.com.
12. Gonthier G (1988) Sémantique et modèles d'exécution des langages réactifs synchrones:
    Application à ESTEREL. PhD thesis, Université d'Orsay, Paris, France (document in French).
13. Berry G (1992) ESTEREL on hardware. In: Mechanized reasoning and hardware design,
    Prentice-Hall Inc, Upper Saddle River, NJ, pp 87–104
14. Potop-Butucaru D, de Simone R (2003) Optimizations for faster execution of ESTEREL programs. In: First ACM and IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'03), Mont Saint-Michel, France, pp 285–315