УДК 004.272.2

АНАЛИЗ ОБЪЕКТНОЙ РЕАЛИЗАЦИИ ЯЗЫКА CAL

Казачёнок М.С.

Научный руководитель – Прихожий А.А., профессор, д.т.н.

Язык CAL (ConcurrentActorLanguage) относится к языкам высокого уровня [1-4]. CAL разработан в Калифорнийском университете Беркли в 2001 году. CAL – это язык потока данных, ориентированный на множество областей применения таких как обработка мультимедиа, сетевая обработка, криптография и т.д. Описание вычисления, представленное в виде диаграммы актеров, соединенных дугами, показывает передачу потоков данных и является хорошим руководством относительно того, может ли поток данных эффективно представлять заданную проблемную область.

Одним из самых простых актеров является актер SimpleActor (см. рисунок 1), который посредством действия action лишь копирует токены со своего входного порта на свой выходной порт.

```
actor SimpleActor () InputPort ==>OutputPort :
action InputPort: [token] ==>OutputPort: [token] end
end
```

Рисунок 1 – Код простого актера SimpleActor

Актер SubActor, представленный на рисунке 2, имеет два входных порта. Как и актер SimpleActor, он также имеет одно действие, но на этот раз действие читает токен с каждого из входных портов. Единственный выходной токен, созданный этим действием, является разницей двух входных токенов.

```
actor SubActor () InPort1, InPort2 ==> Output:
```

action InPort1:[num1],InPort2:[num2]==>[num1–num2] **end**

end

Рисунок 2 – Актер SubActor, вычисляющий разницу двух входных токенов

Пример выполнения актёра SubActor представлен на рисунке 3.

[19, 7, -6], [13, 0, 2] ==> []

-> [7, -6], [0, 2] ==> [6]

-> [-6], [2] ==> [6, 7]

-> [], [] ==> [6, 7, -8]

Рисунок 3 – Пример выполнения актера SubActor

До этого момента у всех актеров было одно действие, однако актер может иметь любое количество действий, в том числе вообще никаких. Пример актера SimpleAndSubActor с двумя действиями представлен на рисунке 4.

actorSimpleAndSubActor () In1, In2 ==> Out1, Out2:

action In1:[num1] ==> Out1: [num1] **end**

action In1:[num1],In2:[num2]==>Out2[num1-num2] **end**

end

Рисунок 4 – Актер Simple, выполняющий два действия

Во многих случаях мы хотим указать дополнительные критерии, которые должны быть выполнены для действия, которое нужно выполнить. Одним из критериев является условие, которое зависит от значений тоукенов или от состояния актера, или одновременно от тоукенов и состояния. Эти условия

могут быть указаны с помощью защитного выражения (guard), как, например, в актёре SortActor представленного на рисунке 5.

```
actorSortActor () InPort ==> Out1, Out2:
action [num] ==> Out1: [num]
guardnum>= 0 end
action [num] ==> Out2: [num]
guardnum< 0 end
end
```

Рисунок 5 – Актер SortActor, использующий защитное выражение

Пример выполнения актера SortActor дан на рисунке 6.

```
[7, -11, 0] ==> [], []
-> [-11, 0] ==> [7], []
-> [0]==> [7], [-11]
-> [] ==> [7, 0], [-11]
```

Рисунок 6 – Пример выполнения SortActor

До сих пор ни у актеров не было действия, которое могло бы повлиять на последующие действия этого актера. Используя переменные состояния, срабатывание действия (action) может сохранить информацию для последующих срабатываний того же самого или другого действия одного и того же актера. Простой пример этого – актер StateActor, представленный на рисунке 7.

```
actor StateActor() Input ==> Output:
sum:=0
action [num] ==> [sum] do
sum:=sum + a;
end
end
```

Рисунок 7 – Актер StateActor, имеющий внутреннее состояние

Пример выполнения актера StateActor дан на рисунке 8.

```
[1, 3, 5] ==> []
-> [3, 5] ==> [1]
-> [5]==> [1, 4]
-> [] ==> [1, 4, 9]
```

Рисунок 8 – Пример выполнения актёра StateActor

Для объектной реализации языка CAL будет созданы следующие классы:

1. класс Actor – будет эмулировать работу актера;
2. класс Interaction – данный класс будет отвечать за взаимодействие актеров, т.е. отвечать за связь входных и выходных портов актеров;
3. класс Port – для сохранения токенов.

Литература

1. G. A. Agha. ACTORS: A Model of Concurrent Computation in Distributed Systems. The MIT Press Series in Artificial Intelligence. MIT Press, Cambridge, 1986.

2. Ernesto Wandeler. Static Analysis of Actor Networks. University of California at Berkeley Berkeley, CA 94720. March 2003.

3. CALActorLanguage [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/CAL_Actor_Language. (Датаобращения: 13.04.2019).

4. CAL Language Report: Specification of the CAL actor language, Johan Eker and Jörn W. Janneck, Technical Memorandum No. UCB/ERL M03/48, University of California, Berkeley, CA, 94720, USA, December 1, 2003