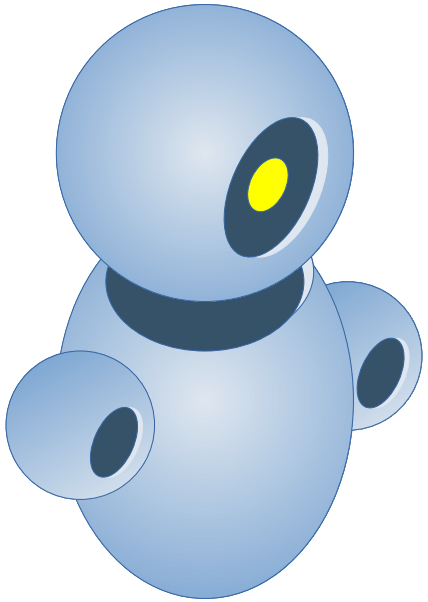
Introduction to Distributed and Embedded Multi-agent Systems

Carlos Eduardo Pantoja[1]
Nilson Mori Lazarin[1,2]

1. Centro Federal de Educação Tecnológica (CEFET/RJ) - 2. Universidade Federal Fluminense (UFF), Brasil

02/2024

Jason StdLib



.print(*parameters*)

- used for printing messages to the console where the system is running.
- It receives any number of parameters, which can be not only strings but also any AgentSpeak term (including variables).

smith Jason

```
1   message("never send a human to do a machine's job!").
2
3   !start.
4
5   +!start: message(M) <-
6           .print("I talked more than ",1," time, ", M).
```

```
[ChonOS EmbeddedMAS] Starting the Multi-Agent System.
Jason Http Server running on http://127.0.1.1:3272
[smith] I talked more than 1 time, never send a human to do a machine's job!
```

https://jason-lang.github.io/api/jason/stdlib/print.html

.my_name(*atom* | *VAR*)

- gets the agent's unique identification in the multi-agent system.
- This identification is given by the runtime infrastructure of the system.

smith | Jason

```
1  !taskTo(jones).
2
3  +!taskTo(Who): .my_name(Who) <-
4          .print("Hello, I'm ", Who,". Executing the task!").
5
6  -!taskTo(Who) <-
7          .my_name(NAME);
8          .print("Sorry, I'm not ", Who, ". I'am ", NAME,"!").
```

```
[ChonOS EmbeddedMAS] Starting the Multi-Agent System.
Jason Http Server running on http://127.0.1.1:3272
[smith] Sorry, I'm not jones. I'am smith!
```

https://jason-lang.github.io/api/jason/stdlib/my_name.html

.random(*VAR*)

- unifies VAR with a random number between 0 and 1.

bob | Jason

```
1  !sortingRandomNumber(3).
2
3  +!sortingRandomNumber(Round): Round>0 <-
4      .random(N);
5      .print("This is a RANDOM number: ", N);
6      !sortingRandomNumber(Round-1).
7
8  -!sortingRandomNumber(Round).
```

```
[ChonOS EmbeddedMAS] Starting the Multi-Agent System.
Jason Http Server running on http://127.0.1.1:3272
[bob] This is a RANDOM number: 0.41921166808754884
[bob] This is a RANDOM number: 0.41014764451156094
[bob] This is a RANDOM number: 0.27646870349394936
```

https://jason-lang.github.io/api/jason/stdlib/random.html

.random(*[t1,t2, ... ,tn],VAR*)

- unifies VAR with a random value from the list.

alice.asl

```
1   listOfNumbers([1,2,3,4,5,6,7,8,9,0]).
2
3   !choosingNUmbersInAList(3).
4
5   +!choosingNUmbersInAList(Round): listOfNumbers(List) & Round > 0 <-
6       .random(List,N);
7       !choosed(Round,N).
8
9   +!choosed(Round,N): numberChoosed(N) <-
10      !choosingNUmbersInAList(Round).
11  +!choosed(Round,N): not numberChoosed(N) <-
12      .print("This is a choosed number of the list: ", N);
13      +numberChoosed(N);
14      !choosingNUmbersInAList(Round-1).
15
16  -!choosingNUmbersInAList(Round).
```

```
[alice] This is a choosed number of the list: 3
[alice] This is a choosed number of the list: 6
[alice] This is a choosed number of the list: 9
```

https://jason-lang.github.io/api/jason/stdlib/random.html

.stopMAS | .stopMAS(*Delay*) | .stopMAS(*Delay*,*sucess*|*fail*)

- aborts the execution of all agents in the multi-agent system.
- return 0 in the main function means that the program executed successfully.
- return 1 in the main function means that the program does not execute successfully and there is some error.

https://jason-lang.github.io/api/jason/stdlib/stopMAS.html

alice.asl

```
1  options([stopRightNow,stopAfter,stopWithError]).
2
3  !start.
4  +!start: options(L) <- .random(L,Decision); !execute(Decision).
5  +!show(M) <- .print(M); !show(M).
6
7  +!execute(stopWithError) <- !!show("Something is very wrong!!!!"); .stopMAS(0,1).
8  +!execute(stopRightNow) <- !!show("Goodbye Cruel World!"); .stopMAS.
9  +!execute(stopAfter) <- !!show("I still living!"); .stopMAS(250).
```

```
nilson@dell:~/chonGroup/distributedAndEmbeddedAI/course/08-GoalsAndPlans/Examples/internalActions/stopMAS/
stopMAS$ jason stopMAS.mas2j
[alice] Something is very wrong!!!!

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':run'.
> Process 'command '/usr/lib/jvm/java-17-openjdk-amd64/bin/java'' finished with non-zero exit value 1
```

```
nilson@dell:~/chonGroup/distributedAndEmbeddedAI/course/08-GoalsAndPlans/Examples/internalActions/stopMAS/
stopMAS$ jason stopMAS.mas2j
[alice] Goodbye Cruel World!
nilson@dell:~/chonGroup/distributedAndEmbeddedAI/course/08-GoalsAndPlans/Examples/internalActions/stopMAS/
stopMAS$ jason stopMAS.mas2j
[alice] I still living!
[alice] I still living!
[alice] I still living!
```

.wait(*miliseconds*) | .wait(*Predicate*)

- suspend the intention for the time specified by in milliseconds
- Suspend the intention until the *Predicate* is added in the belief base.

trinity | Jason

```
1   timeSpent(0).
2
3   !clock.
4   !alarm(3).
5
6   +!clock: timeSpent(T) <- .wait(1000);
7           -+timeSpent(T+1); .print("The time is: ", T+1);  !clock.
8
9   +!alarm(A): A \== stop <-
10          .wait(timeSpent(A)); !!display("Wake up, Neo..."); +wakeUp;
11          .wait(alarm(off));  .print("Good morning Neo..."); .stopMAS.
12
13  +!display(M): not alarmf(off)<- .wait(1000); .print(M); !display(M).
14
15  +wakeUp <- .random(R); .wait(5000*R); +alarm(off).
```

```
[ChonOS EmbeddedMAS] Starting the Multi-Agent System.
Jason Http Server running on http://127.0.1.1:3272
[trinity] The time is: 1
[trinity] The time is: 2
[trinity] The time is: 3
[trinity] Wake up, Neo...
[trinity] The time is: 4
[trinity] Wake up, Neo...
[trinity] The time is: 5
[trinity] Wake up, Neo...
[trinity] Good morning Neo...
[trinity] The time is: 6
```

https://jason-lang.github.io/api/jason/stdlib/wait.html

.count(*predicate(term,_),Unify*)

- counts the number of occurrences of a particular belief (pattern) in the agent's belief base.

bob | Jason

```
1   day(businessDay, monday).
2   day(businessDay, tuesday).
3   day(businessDay, wednesday).
4   day(businessDay, thursday).
5   day(businessDay, friday).
6   day(weekend, saturday).
7   day(weekend, sunday).
8
9   !start.
10
11  +!start <-
12          .count(day(businessDay,_),Total1);
13          .count(day(weekend,_),Total2);
14          .print("My week has ", Total1, " business days and ", Total2, " in the weekend!").
```

```
[ChonOS EmbeddedMAS] Starting the Multi-Agent System.
Jason Http Server running on http://127.0.1.1:3272
[bob] My week has 5 business days and 2 in the weekend!
```

https://jason-lang.github.io/api/jason/stdlib/count.html

.min(*List,Unify*) | .max(*List,Unify*)

bob | Jason

```
1   listOfNumbers([1,2,3,4,5,6,7,8,9,0]).
2
3   !start.
4   !anyNumber(8).
5
6   +!start: listOfNumbers(List) <-
7           .min(List,Min); .max(List,Max);
8           .print("The min is: ", Min ," and the max is: ", Max).
9
10  +!anyNumber(N): listOfNumbers(List) & .max(List,N) <- .print("The number ", N, " is the bigest in the list").
11
12  +!anyNumber(N): listOfNumbers(List) & .min(List,N) <- .print("The number ", N, " is the smallest in the list").
13
14  -!anyNumber(N) <- .print(N, " isn't nether biggest or smallest").
```

- gets the maximum value within a list of terms.

```
[ChonOS EmbeddedMAS] Starting the Multi-Agent System.
Jason Http Server running on http://127.0.1.1:3272
[bob] 8 isn't nether biggest or smallest
[bob] The min is: 0 and the max is: 9
```

https://jason-lang.github.io/api/jason/stdlib/min.html
https://jason-lang.github.io/api/jason/stdlib/max.html

Agradecimentos

WESAAC 2022

OBRIGADO!

pantoja@cefet-rj.br
nilson.lazarin@cefet-rj.br