RALD-LARD MACHINES

Models of computation exam

Worou Akiyo

aworou@student.ethz.ch

aworou | 20-909-594

Model definition

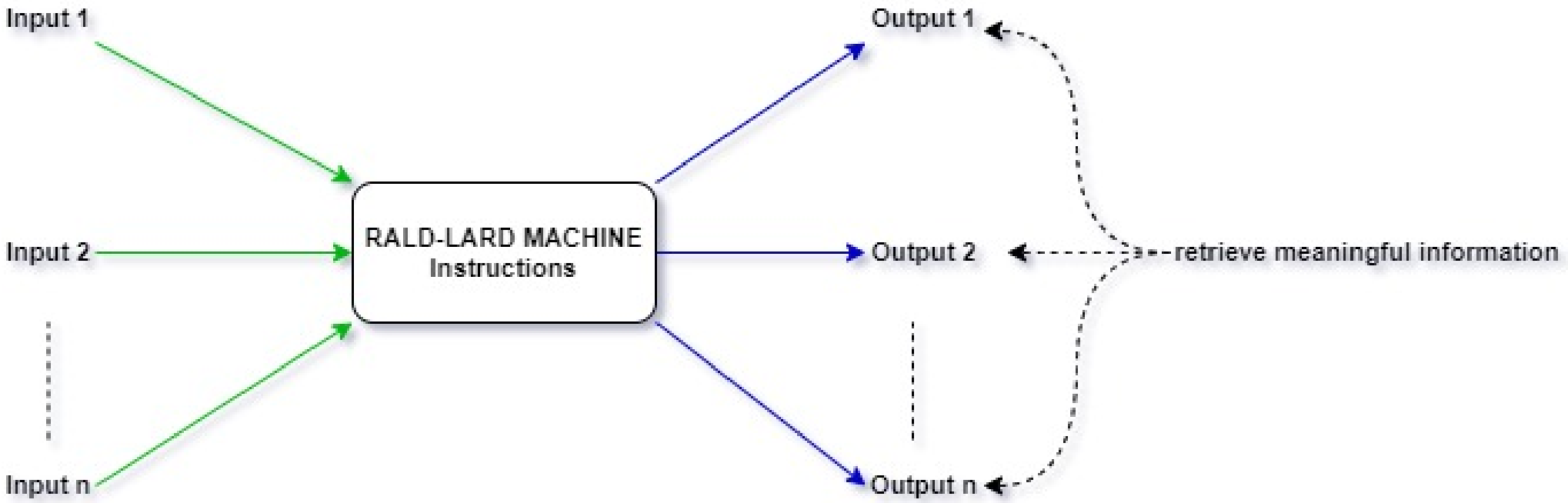
Input and output modalities

- Input modality : O$\underbrace{CCCC\ldots CC}_{n\geq 0}$ or $\underbrace{CCCC\ldots CC}_{n\geq 0}$O
- Output modality : $\underbrace{CCCC\ldots CC}_{n\geq 0}O\underbrace{CCCC\ldots CC}_{m\geq 0}$
- Meaningful output : $\underbrace{CCCC\ldots CC}_{n\geq 0}$O or O$\underbrace{CCCC\ldots CC}_{n\geq 0}$
- O : delimiter and C : repeated character

Rules

- Add and delete C on the right or on the left of the delimiter for a specific input
- If deletion is impossible the machine will stop
- Take set of instructions executed in a loop
- The instructions are assigned to a specific input

Model representation

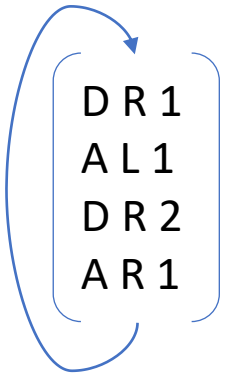


Model instructions

- One line of instructions : Delete/Add Left/Right input id
- To make things easier let denote
  - D:= Delete
  - A:=Add
  - L:=Left
  - R:=Right
- Example : D L 1 ===> Delete C on the left of O for the first input

Model loop of instructions

- Input 1 : OCC
- Input 2 : OC



Steps	Instructions	Input 1	Input 2
1	D R 1	OC	OC
2	A L 1	COC	OC
3	D R 2	COC	O
4	A R 1	COCC	O
5	D R 1	COC	O
6	A L 1	CCOC	O
7	D R 2	CCOC	halt

Properties

- When there is only one input the model is called mono RALD-LARD machine and the input id can be omitted in the instructions

- Validity : A set of instructions is valid if there at least one meaningful output

- Halting : The machine will always halt if in the instructions there is more deletion than appending for at least one input

Computational processes in action

Mono RALD-LARD Machines

Compute n+1 and n-1

- Input : O<u>CCC…CCC</u>
  n
- Ouput : O<u>CCC…CCCC</u>
  n+1
- Instructions : (A R / D L)
- Steps : 1

Input
OCCCC
OCCCCC
halt

- Input : O<u>CCC…CCC</u>
- Ouput : O<u>CCC…CCCC</u>
  n-1
- Instructions : (D R / D L)
- Steps : 1

Input
OCCCC
OCCC
halt

Compute n//2

- Input : O$\underbrace{CCC...CCC}_{n}$
- Ouput : $\underbrace{CCC...CC}_{n//2}$O
- Instructions :

D R
D R
A L

- Steps : 3n//2
- Bonus : n//m is possible with

D R
D R
…
…
D R
A L

m

Input	Input
OCCCC	OCCC
OCCC	OCC
OCC	OC
COCC	COC
COC	CO
CO	halt
CCO	
halt	

Compute 2n

- Input : O$\underbrace{CCC...CCC}_{n}$
- Ouput : $\underbrace{CCC...CCCCCCC}_{2n}$O
- Instructions :

D R
A L
A L

- Steps : 3n
- Bonus : mxn is possible with

D R
A L
A L
…
…
A L

m

Input
OCCC
OCC
COCC
CCOCC
CCOC
CCCOC
CCCCOC
CCCCO
CCCCCO
CCCCCCO
halt

Parity

- Input : O$\underbrace{CCC\ldots CCC}_{n}$
- Ouput : CO if n odd and O else
- Instructions :

D R
A L
D R
D L

- Steps : 2n
- Bonus : check if n is divisible by m

D R
A L
D R
D R
…
…
D R
D L

m-1

Input	Input
OCCCC	OCCC
OCCC	OCC
COCCC	COCC
COCC	COC
OCC	OC
OC	O
COC	CO
CO	halt
O	
halt	

Computational processes in action

RALD-LARD Machines with 2 inputs

Compute n+m

- Input 1 : O$\underbrace{CCC...CCC}_{n}$
- Input 2 : O$\underbrace{CCC...CCC}_{m}$
- Outputs :
  - Output 1 : O$\underbrace{CCC...CCC}_{n+m}$ | Output 2 : O
- Instructions :

```
D R 2
A R 1
```

- Steps : 2m
- Tips : Choosing input 2 such as m≤n

Input 1	Input 2
OCCC	OCC
OCCC	OC
OCCCC	OC
OCCCC	O
OCCCCC	O
	halt

Compute n-m

- Input 1 : O$\underbrace{CCC...CCC}_{n}$
- Input 2 : O$\underbrace{CCC...CCC}_{m}$
- Outputs :
  - Output 1 : O$\underbrace{CCC...CCC}_{n-m}$ | Output 2 : O ===> m≤n
  - Output 1 : O | Output 2 : OCC…CCC ===> m>n
- Instructions :

```
D R 2
D R 1
```

- Steps : 2min(n,m)

Input 1	Input 2
OCCC	OCC
OCCC	OC
OCC	OC
OCC	O
OC	O
	halt

Compare n and m

- Input 1 : OCCC…CCC (n C's)
- Input 2 : OCCC…CCC (m C's)
- Outputs :
  - Output 1 : O | Output 2 : O ===> n=m
  - Output 1 : COCCC…CCC or OCC…CCC | Output 2 : O ===> m<n
  - Output 1 : O | Output 2 : COCCC…CCC or OCC…CCC ===> m>n
- Instructions :

```
D R 1
A L 1
D R 2
D L 1
```

- Steps : ≤ 4min(n,m)

Input 1	Input 2
OCCC	OCC
OCC	OCC
COCC	OCC
COCC	OC
OCC	OC
OC	OC
COC	OC
COC	O
OC	O
O	O
CO	O
	halt

Input 1	Input 2
OCC	OCC
OC	OCC
COC	OCC
COC	OC
OC	OC
O	OC
CO	OC
O	O
halt	

RALD-LARD Machines into the world of models

Observations

- Can easily do some basic arithmetic computations
- Cannot do some recursion based computation : mxn for 2 inputs
- Cannot reuse multiple times an input because of the halting definition
- Cannot solve easily solve non-arithmetic based problems
- The model does not have states so it cannot solve state based problems

Comparison with Cyclic tag systems

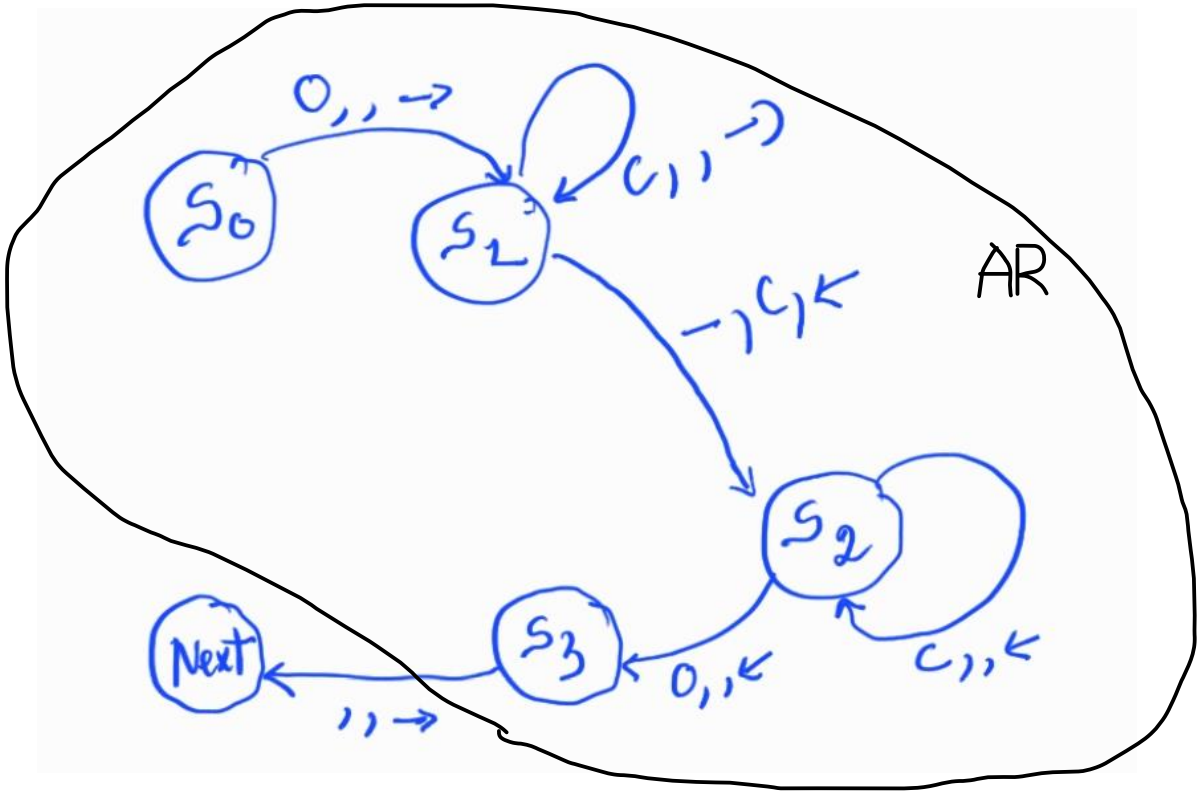
- Similar to cyclic tag system for the instructions loop
- Contrary to cyclic tag it does not delete the odd characters but have the ability to delete
- The transformation is executed at the delimiter position, so the machine only needs to know the closest neighboors if they exist

Simulation of another models

- Cannot simulate one of the models seen in class (at least I was not able to do it)

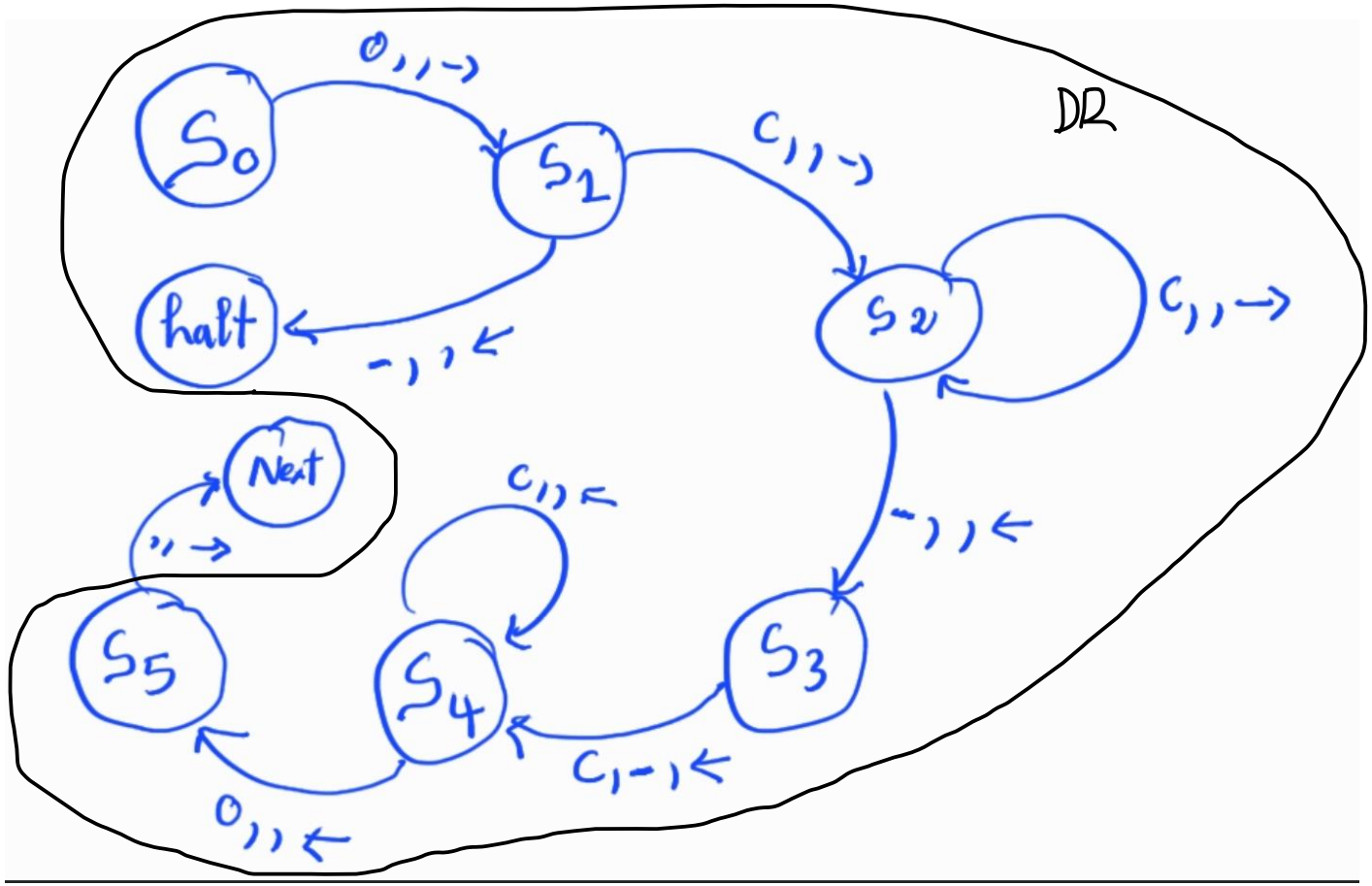
Turing machine that simulates RALD-LARD Machine

- We can consider a TM with input _CCOCC_
- A R can be encoded like :



Turing machine that simulates RALD-LARD Machine

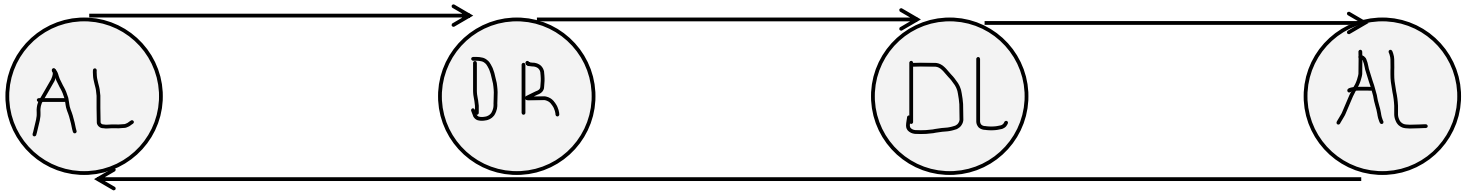
- D R can be encoded like :



Turing machine that simulates RALD-LARD Machines

- We can then replace Next by the next instruction and make the loop

- Example :

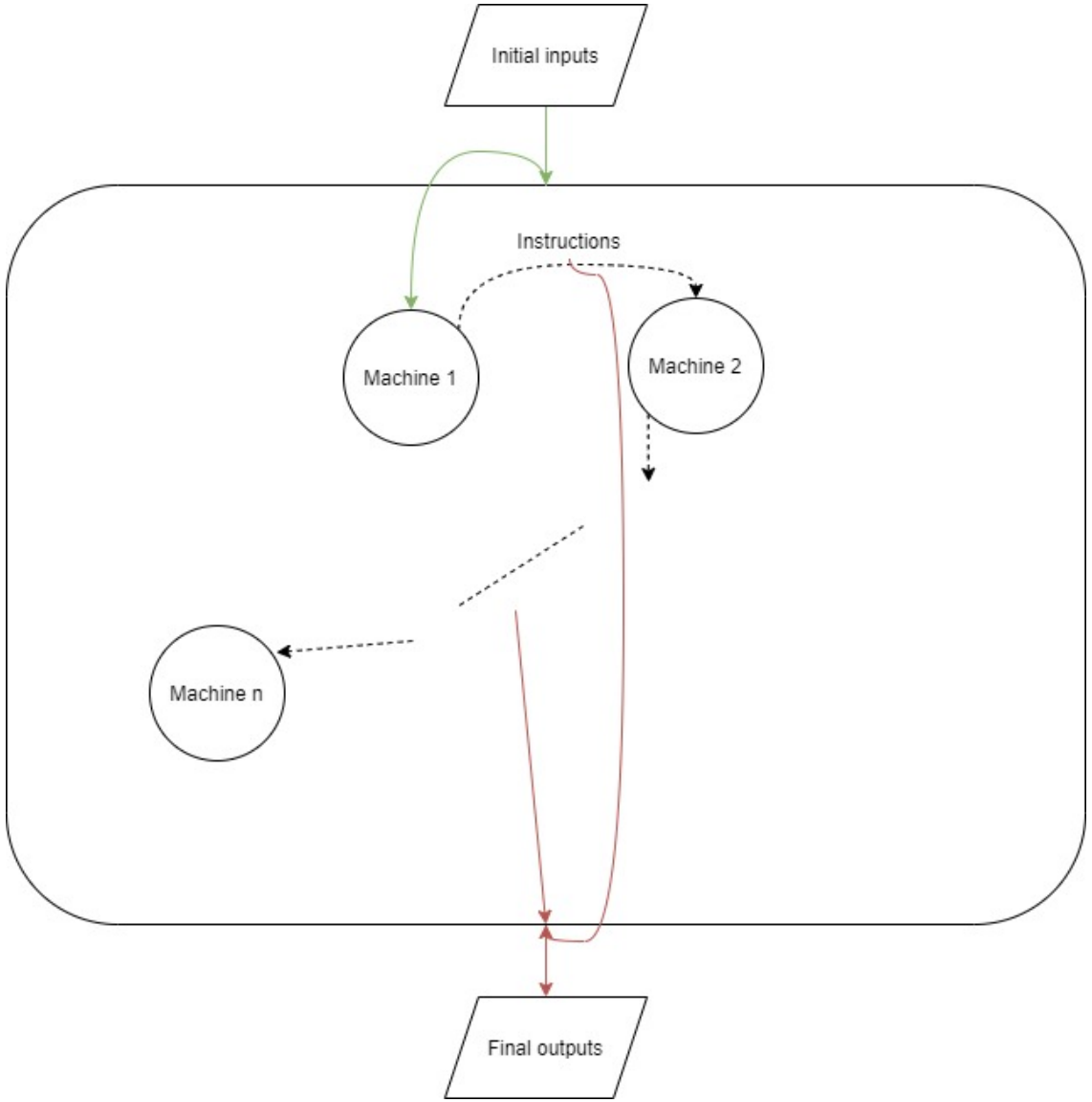


Solving the recursion issue?

RALD-LARD Machines stacking

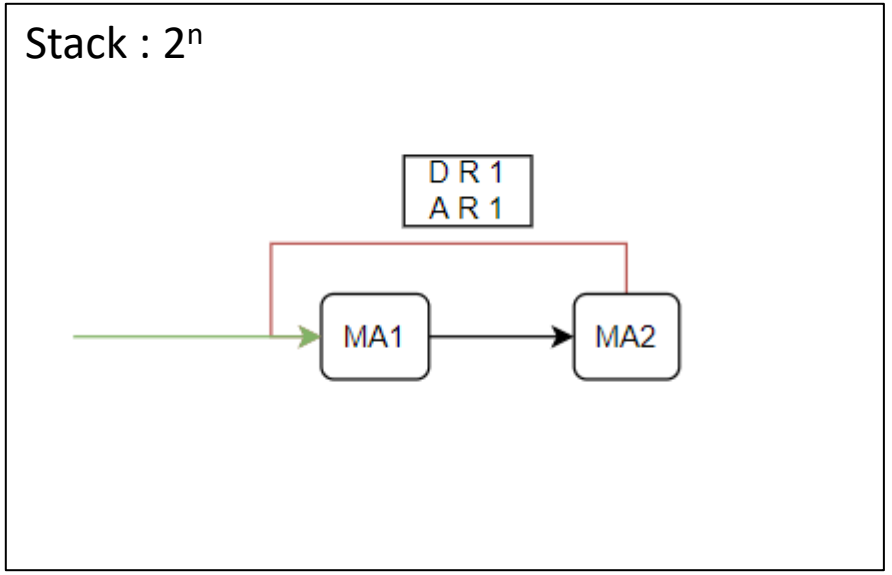
Stacking overview

- Multiple machines
- Simple transition after a machine halts
- Instruction transition : apply a set of instructions, not necessarily in a loop
- When an instruction transition failed the entire model stops


Initial inputs
Instructions
Machine 1
Machine 2
Machine n
Final outputs

Computing 2^n

- Input 1 : OCCC…CCC
- Input 2 : OC ($m^n = 1$)
- Output 1 : O
- Output 2 : OCCC…CCC
- Machine 1 (MA1)$^{2^n}$: compute n – 1
- Machine 2 (MA2) : compute 2m



Conclusion

- Interesting model even if it is a bit limited
- Work on RALD or LARD versions
- Maybe add more possibilities
- Variation in the meaning of the output
- Stacking can be powerful