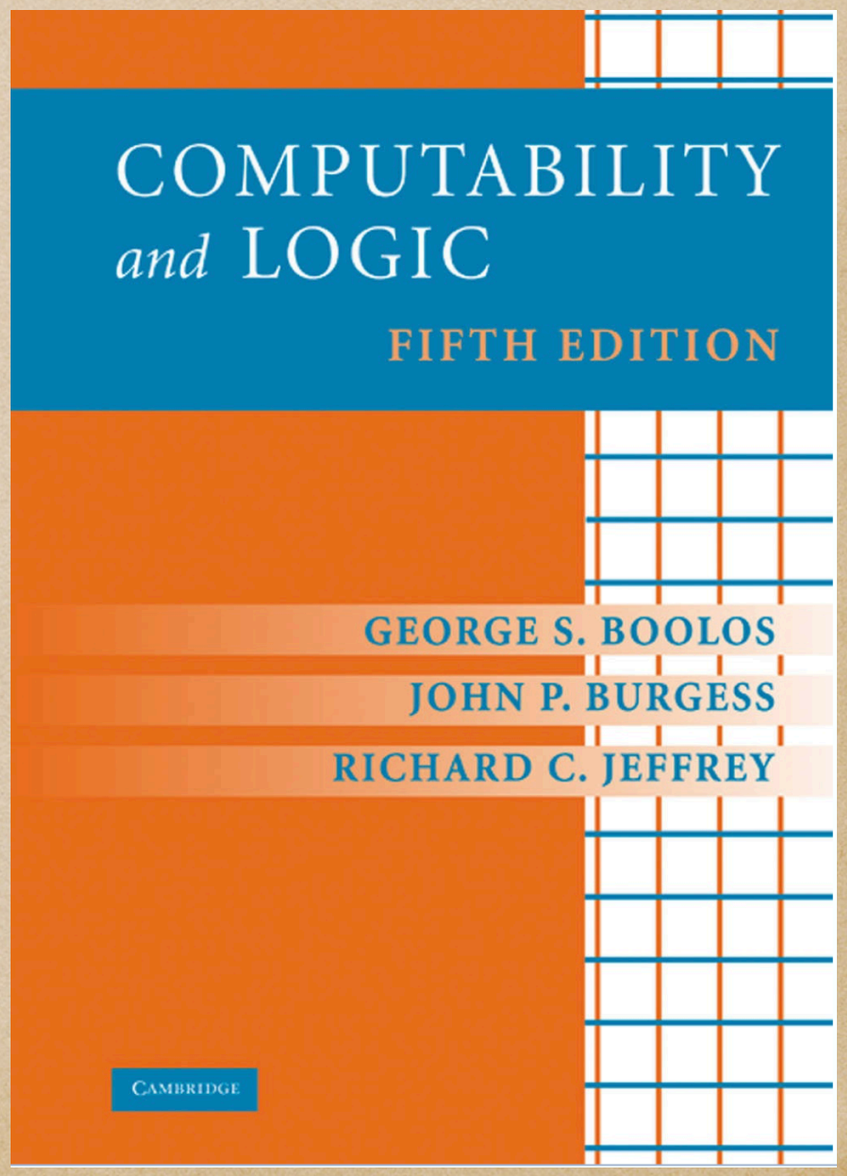
Register Machines and Computable Functions in HOL4

Prensenter: Zhuo (Zoey) Chen zhuo.chen1@anu.edu.au
Supervisor: Michael Norrish michael.norrish@data61.csiro.au

Reference

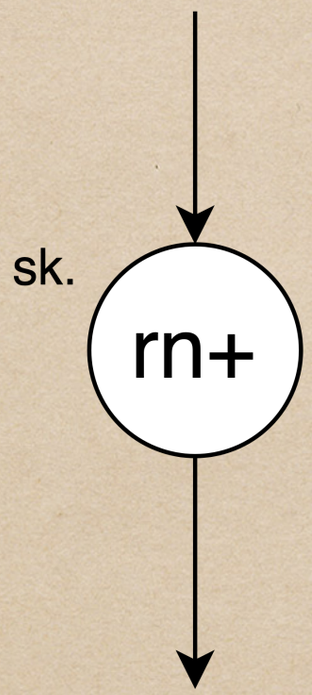
Boolos, G. G.; Burgess, J. P.; and Jeffrey, R. C.
Computability and logic.



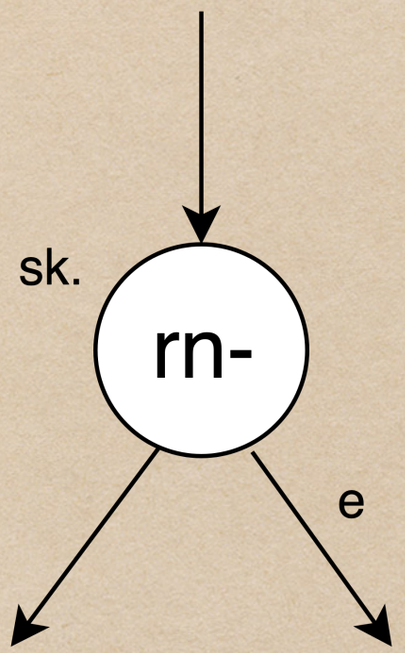
Abacus Machine

Register Machine

Register Machine



Add one stone to the basket n



Come out from arrow e if basket n is empty, otherwise remove one stone from basket n and come out from another arrow

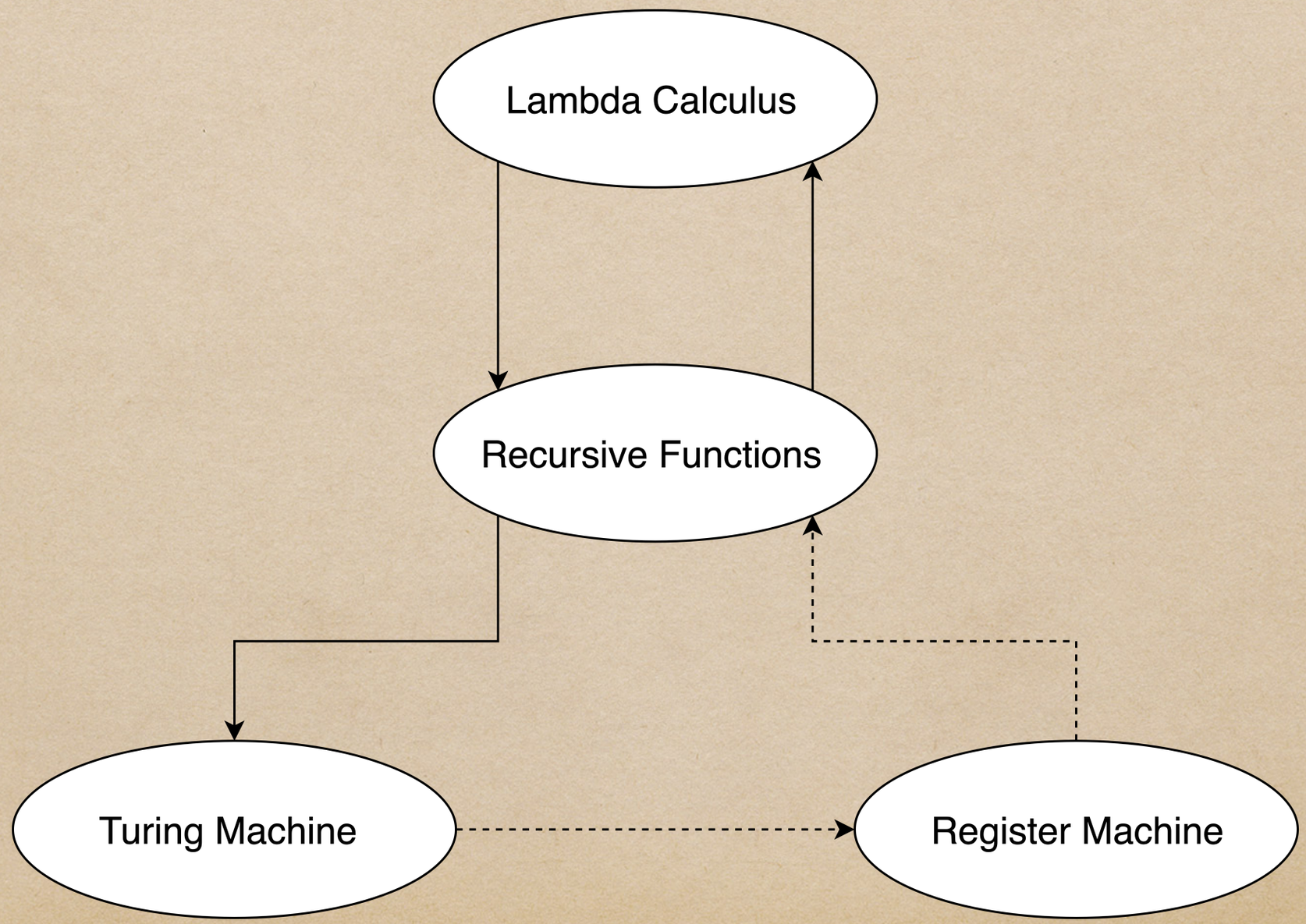
HOL4

- An Interactive Theorem Prover (ITP) written on top of Standard Meta Language
- What is an ITP? - A software tool that does math proof by providing a human computer collaborative service

HOL4

- Tactics: simplify or prove goals (using user provided theorems (optional))
- Record: `p1 = <| Person := "Mary"; Fruits:= ["Apple", "Orange"]; Age := 21 |>`
- Set: `{1; 2; 4}`
- List: `[1;27;5]`
- Function Composition: `(f o g) x = f(g(x))`

Relation between Recursive Functions, Lambda Calculus, Turing Machine and Register Machine



- Build Register Machine Model
- Simple computable functions and proof
- Composition and Primitive Recursion
- Register Machine and Recursive Functions

- Build Register Machine Model
- Simple computable functions and proof
- Composition and Primitive Recursion
- Register Machine and Recursive Functions

Register Machine Model

```
rm = <|
    Q : state set;
    tf : state -> action ;
    q0 : state ;
    In : reg list ;
    Out : reg
|>
```

Register Machine Model

- RUN : walk through the states according to the machine's transition function as well as perform the action inside the current state each time

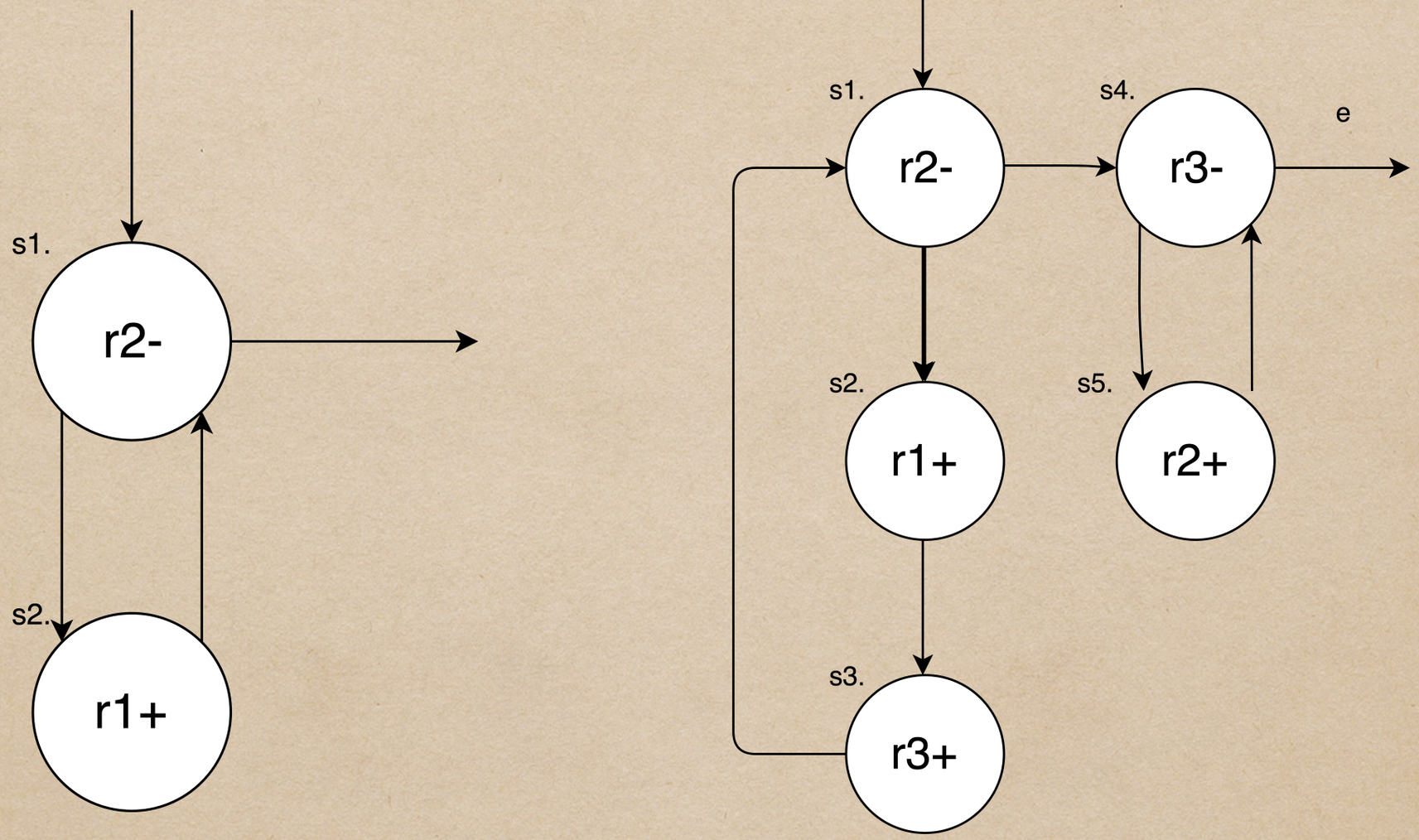
- `wfrm`: (Wellformedness) : finite states, finite registers, initial state(`q0`) is inside state set `Q`, closure

- Build Register Machine Model
- Simple computable functions and proof
- Composition and Primitive Recursion
- Register Machine and Recursive Functions

Simple Computable Functions simulated by Register Machines

- Addition (`simp_add`, `addition`)
- Multiplication (`multiplication`)
- Exponentiation (`exponential`)
- Factorial (`factorial`)
- Projection (`pi`)
- `constant`, `add1`, `identity`, `empty`, `transfer`, `double`, etc

Addition



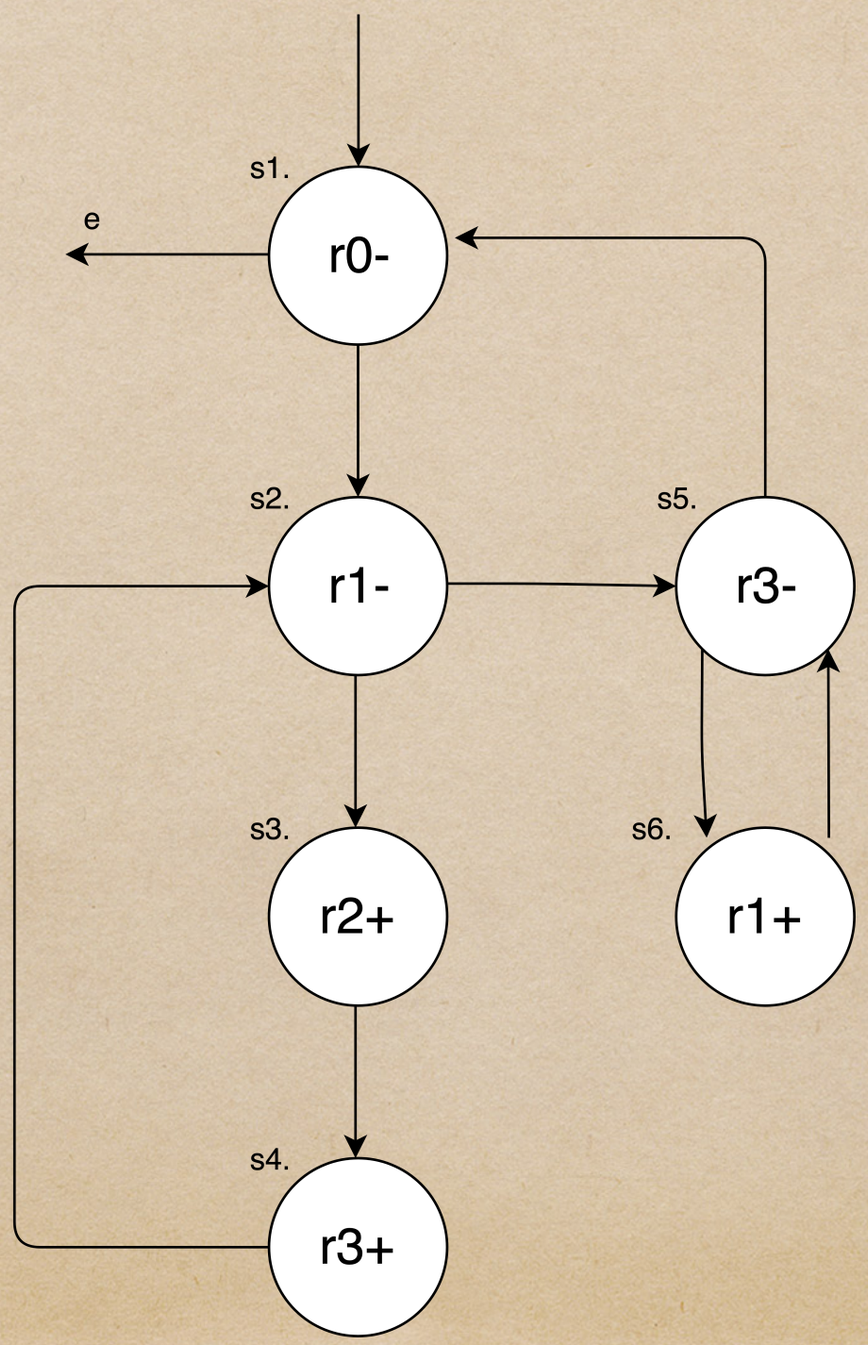
simp_add

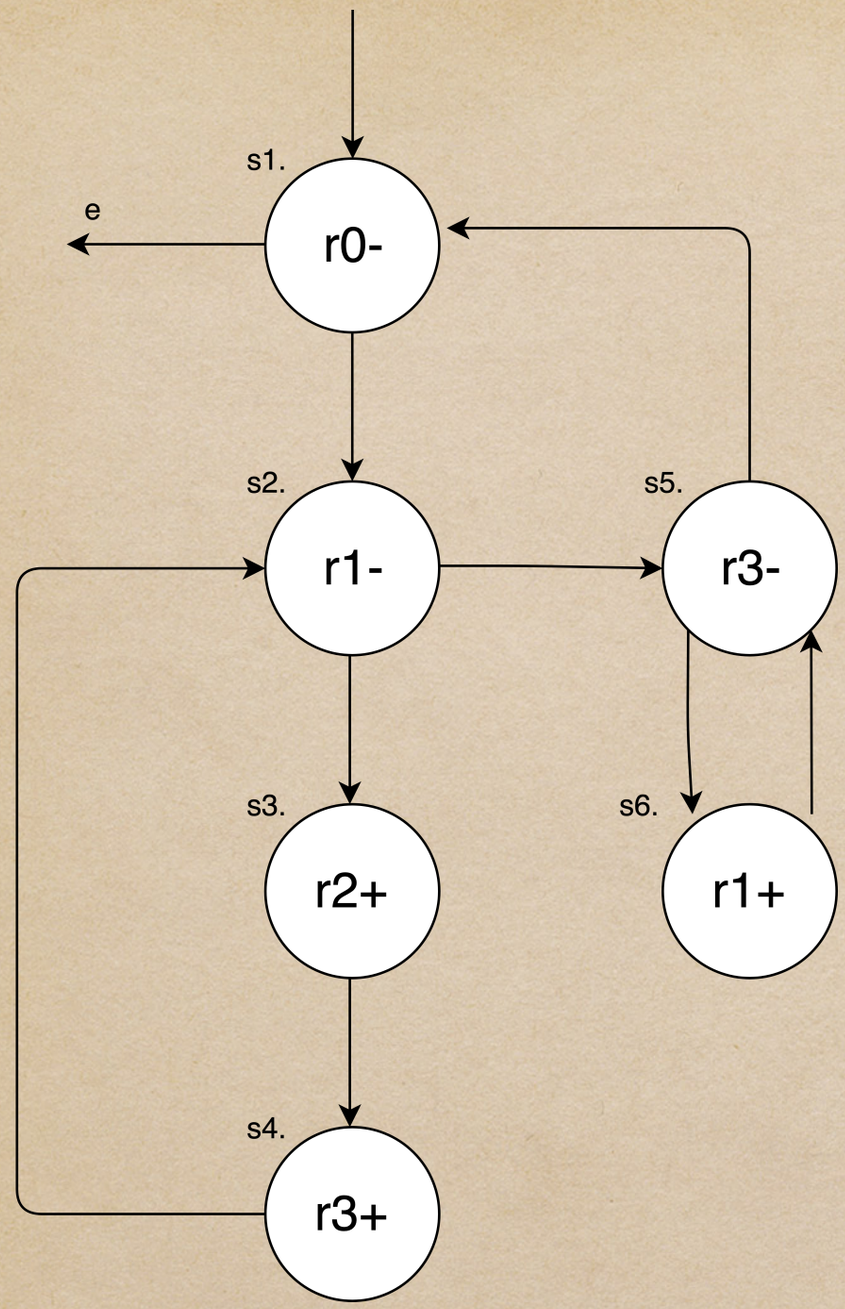
addition

simp_add : PROOF

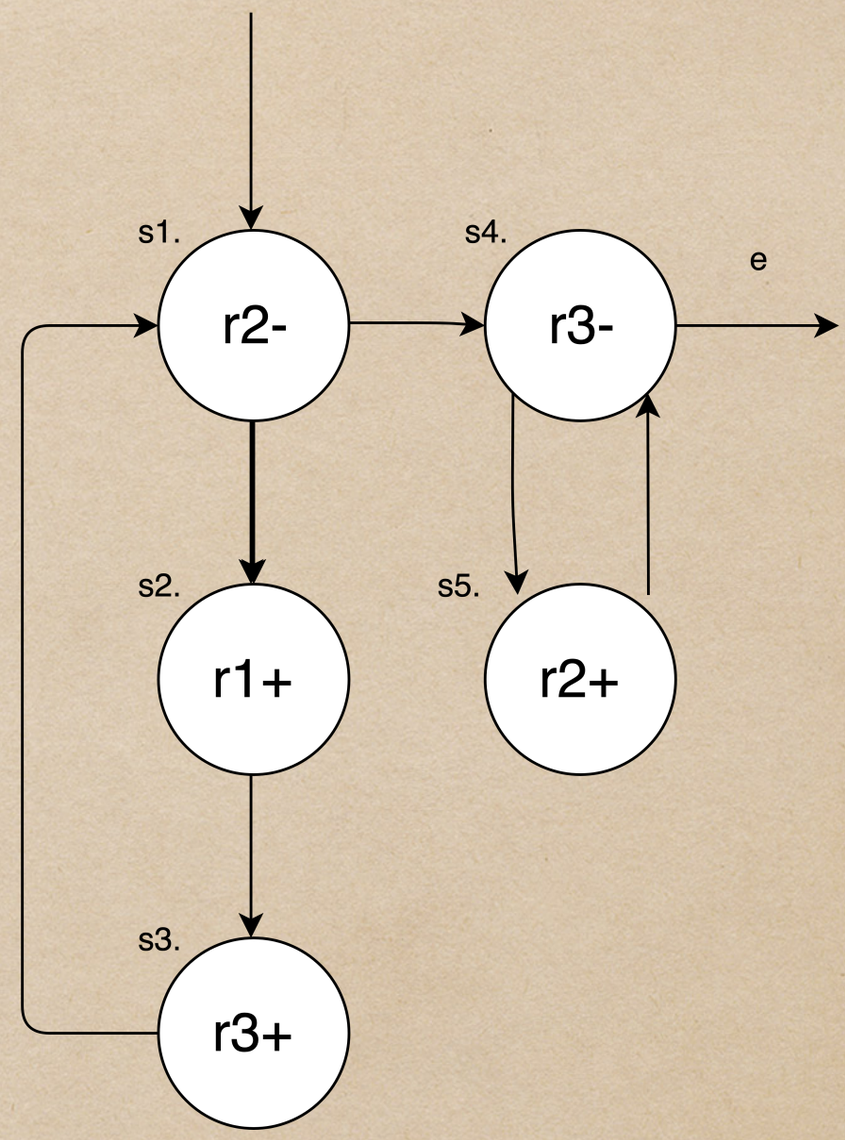
```
Theorem simp_add_correct:
  correct2 (+) simp_add
Proof
  rw[simp_add_def, correct2_def, init_machine_def, run_machine_def, RUN_def]
>>
  qmatch_abbrev_tac `FST (WHILE gd (r m) init) 1 = a + b` >>
  `∀rs0. FST (WHILE gd (r m) (rs0, SOME 1)) 1 = rs0 1 + rs0 2`
    suffices_by rw[Abbr`init`, indexedListsTheory.findi_def] >>
  gen_tac >>
  rw[Abbr`r`, Abbr`m`, Abbr`gd`] >>
  Induct_on `rs0 2` >>
  rw[Ntimes WHILE 2, run_machine_1_def, APPLY_UPDATE_THM]
QED
```

Multiplication



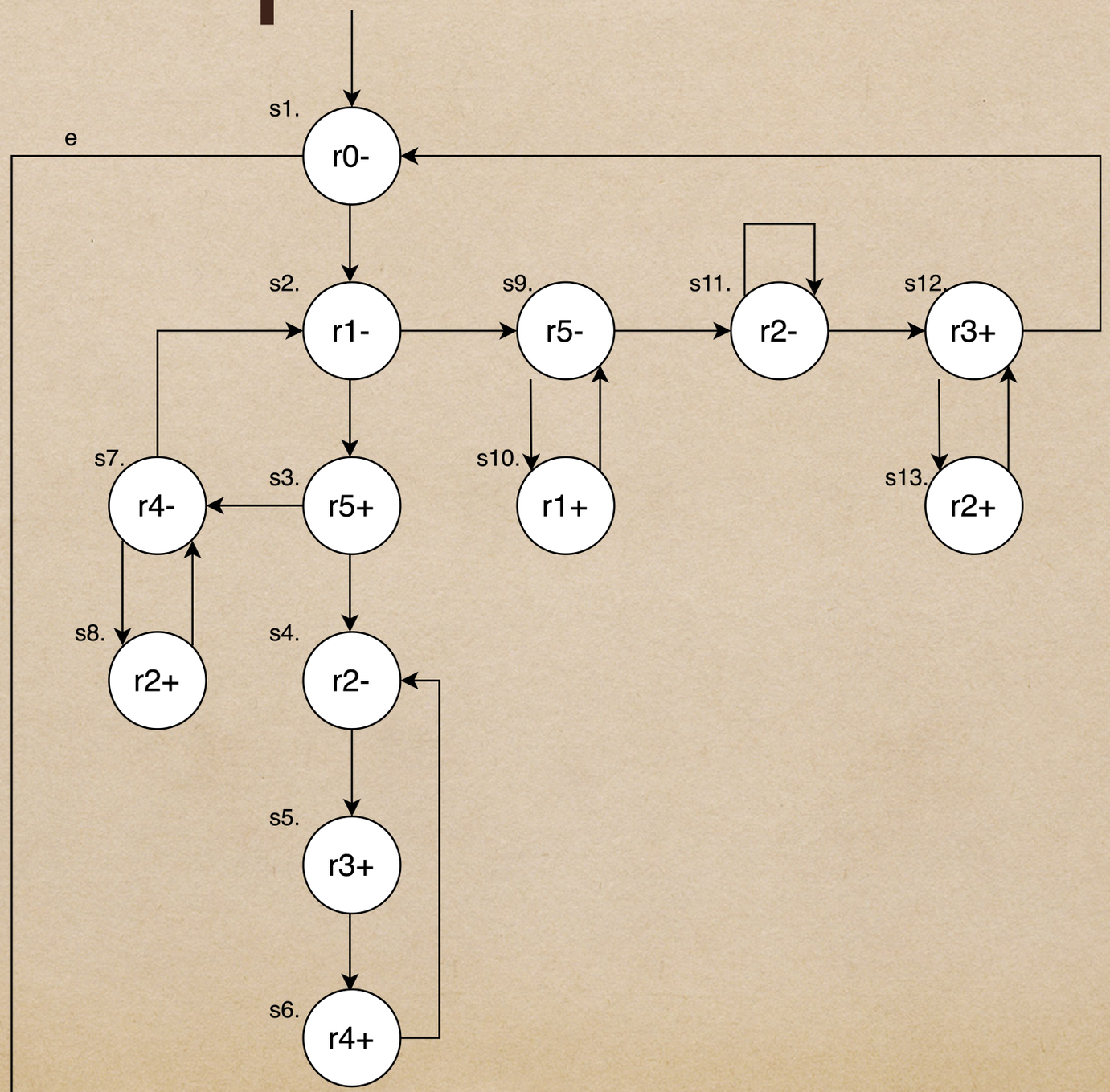


multiplication



addition

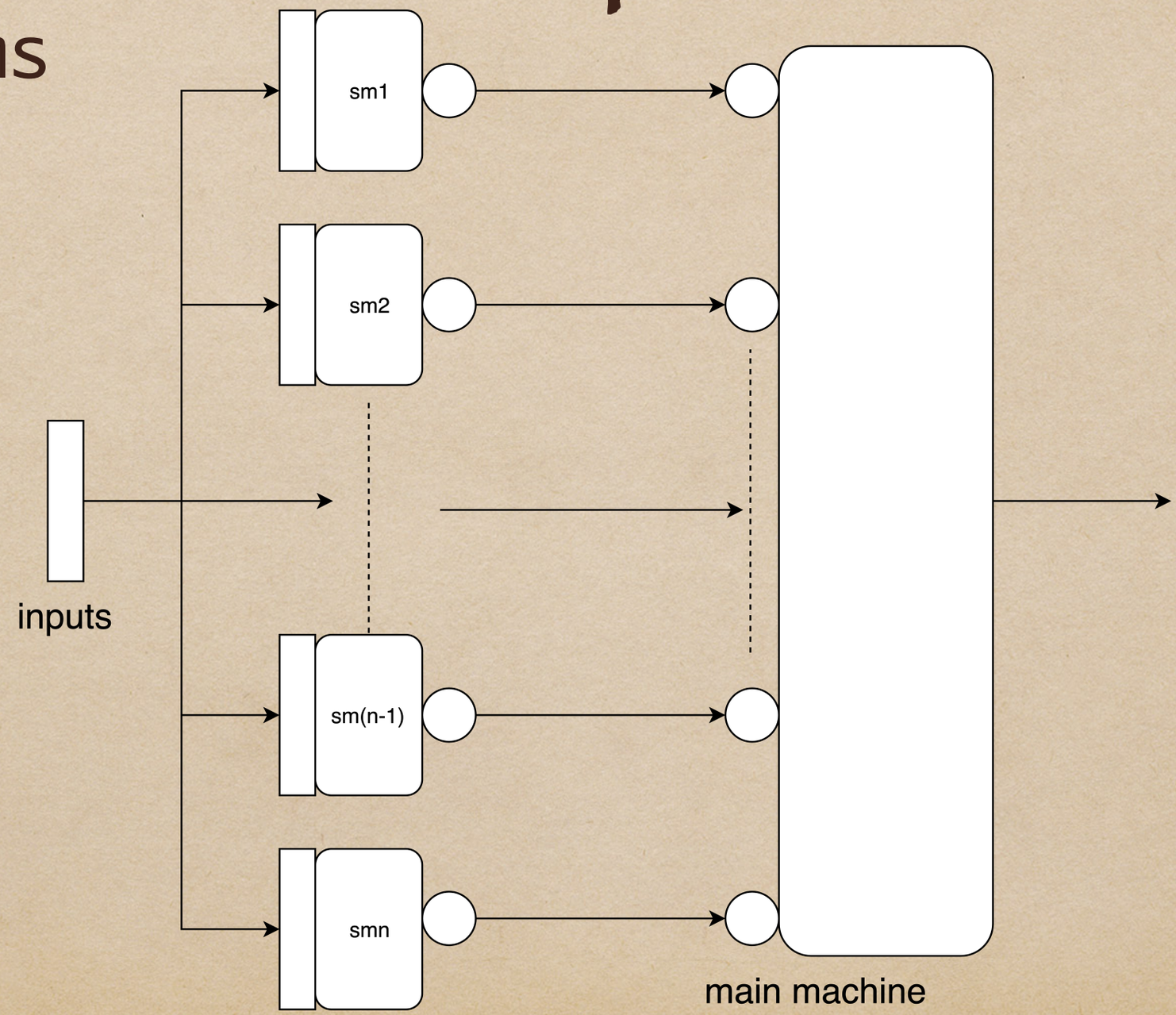
exponential



- Build Register Machine Model
- Simple computable functions and proof
- Composition and Primitive Recursion
- Register Machines and Recursive Functions

Cn (Composition)

Cn m ms



copied inputs | submachine | output

Cn (Composition)

- Problem: Machines might use the same states/registers
- Solution: Rename! (`msInst, mrInst`)
- Input: machine_number, machine
- Output: npair machine_number state/register

  (npair: bijection from N * N to N)

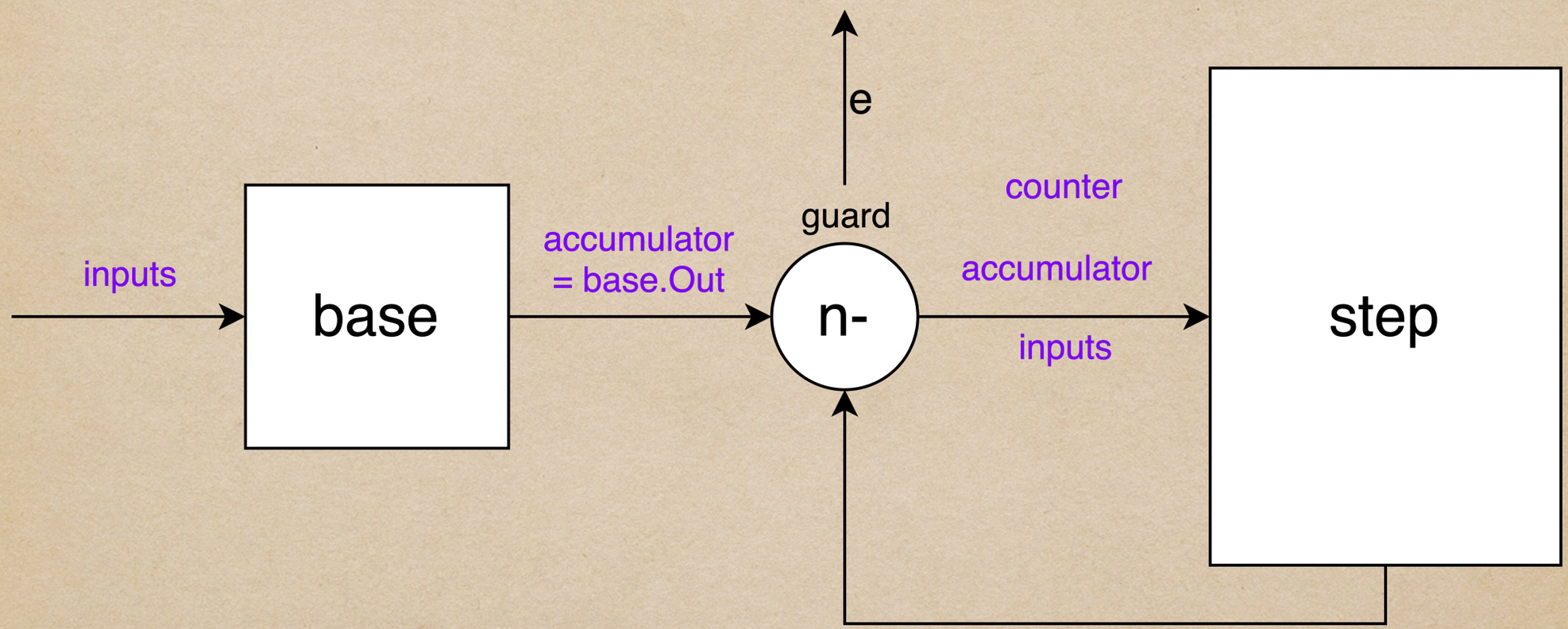
Cn (Composition)

- Problem: Machines are separate, are not aware of the existence of other machines
- Solution: `link` function, `dup0` function

Cn (Composition)

- Defined
- Tested
- Not yet verified

Pr (Primitive Recursion)



- Under development

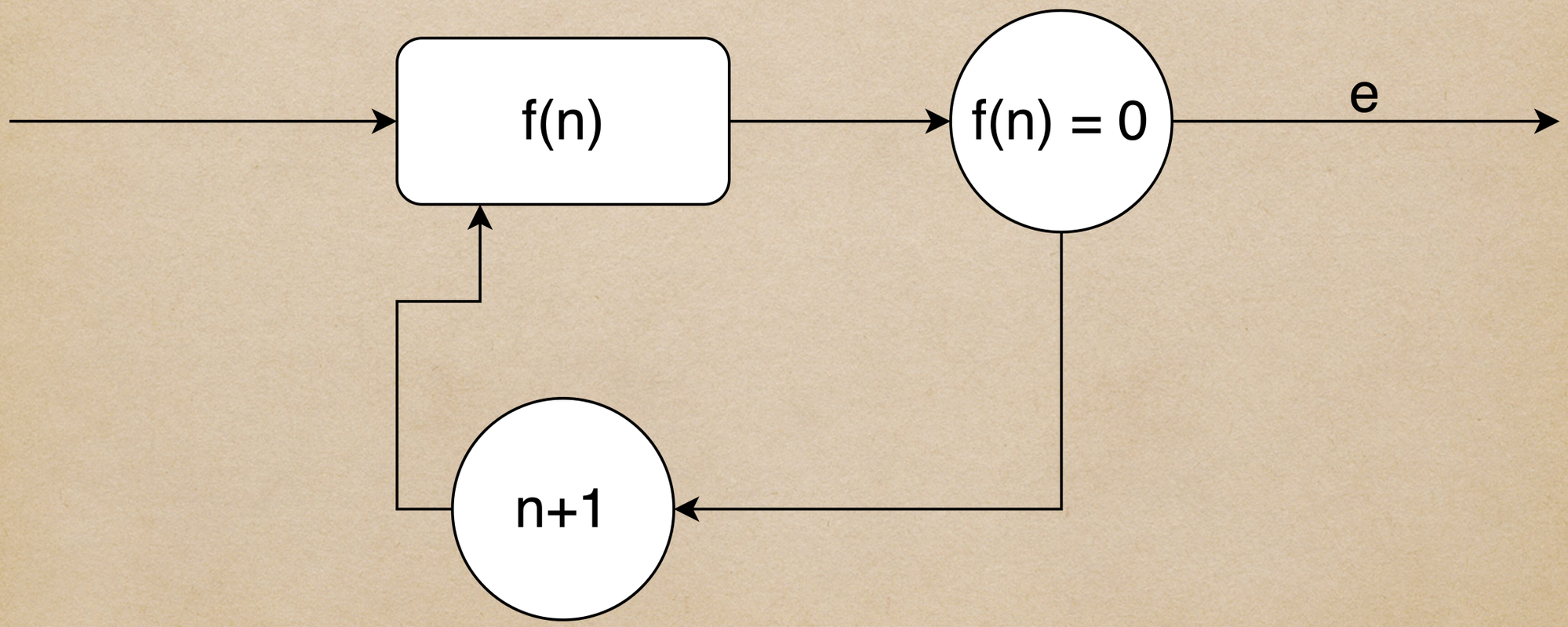
- Build Register Machine Model
- Simple computable functions and proof
- Composition and Primitive Recursion
- Register Machine and Recursive Functions

Recursive Functions

- `O : constant 0`
- `SUC: add1`
- Projection: `Pi m n` (m-th element from list n)
- Composition: `Cn`
- Primitive Recursion: `Pr`
- Minimisation: `Mu f n`

Minimisation

Find the least n such that f(n) = 0



Future Work

- Register Machine to Recursive Functions
  - Finish Primitive Recursion
  - Finish Minimisation
- Programming support to make computation with register machine more user friendly
- Turing Machine to Register Machine
- More proofs

Conclusion

- Build Register Machine Model
- Simple computable functions and proof
- Composition and Primitive Recursion
- Register Machine and Recursive Functions
- Future Work

Great thanks to my supervisor
Michael Norrish !

Thank you for watching ~