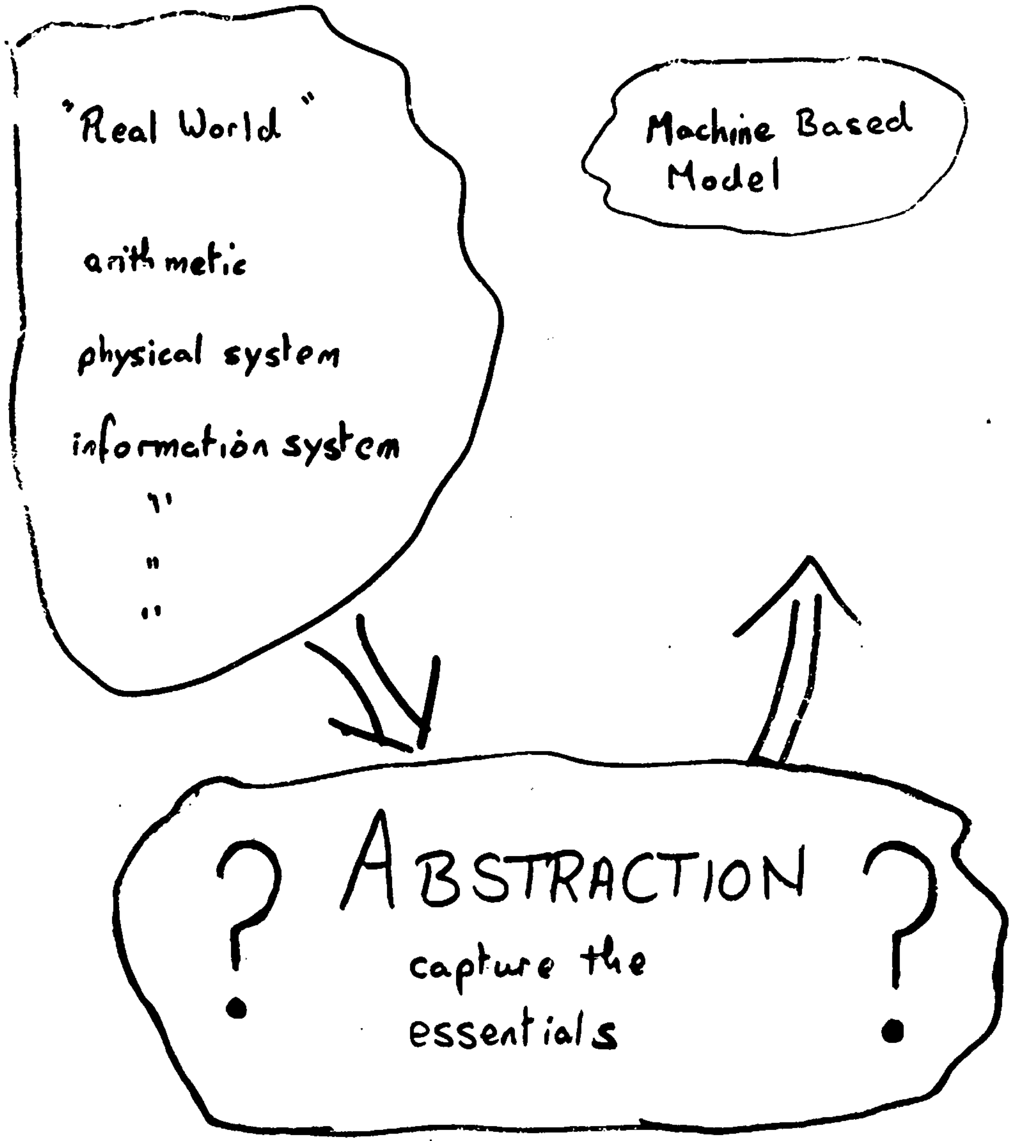
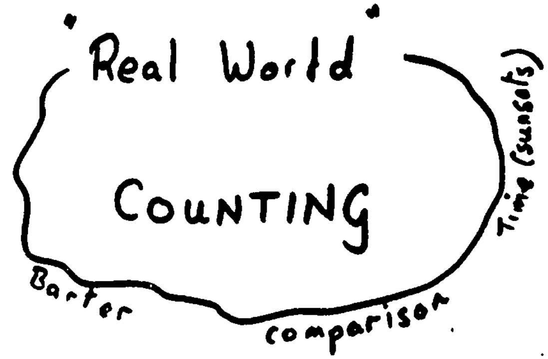
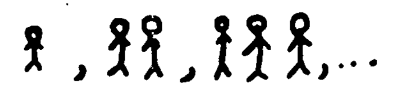


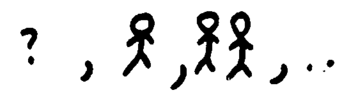


Need

"concept of a number"



"concept of zero"



"concept of successor"

Our "representation"/"model" will need to manifest these concepts.

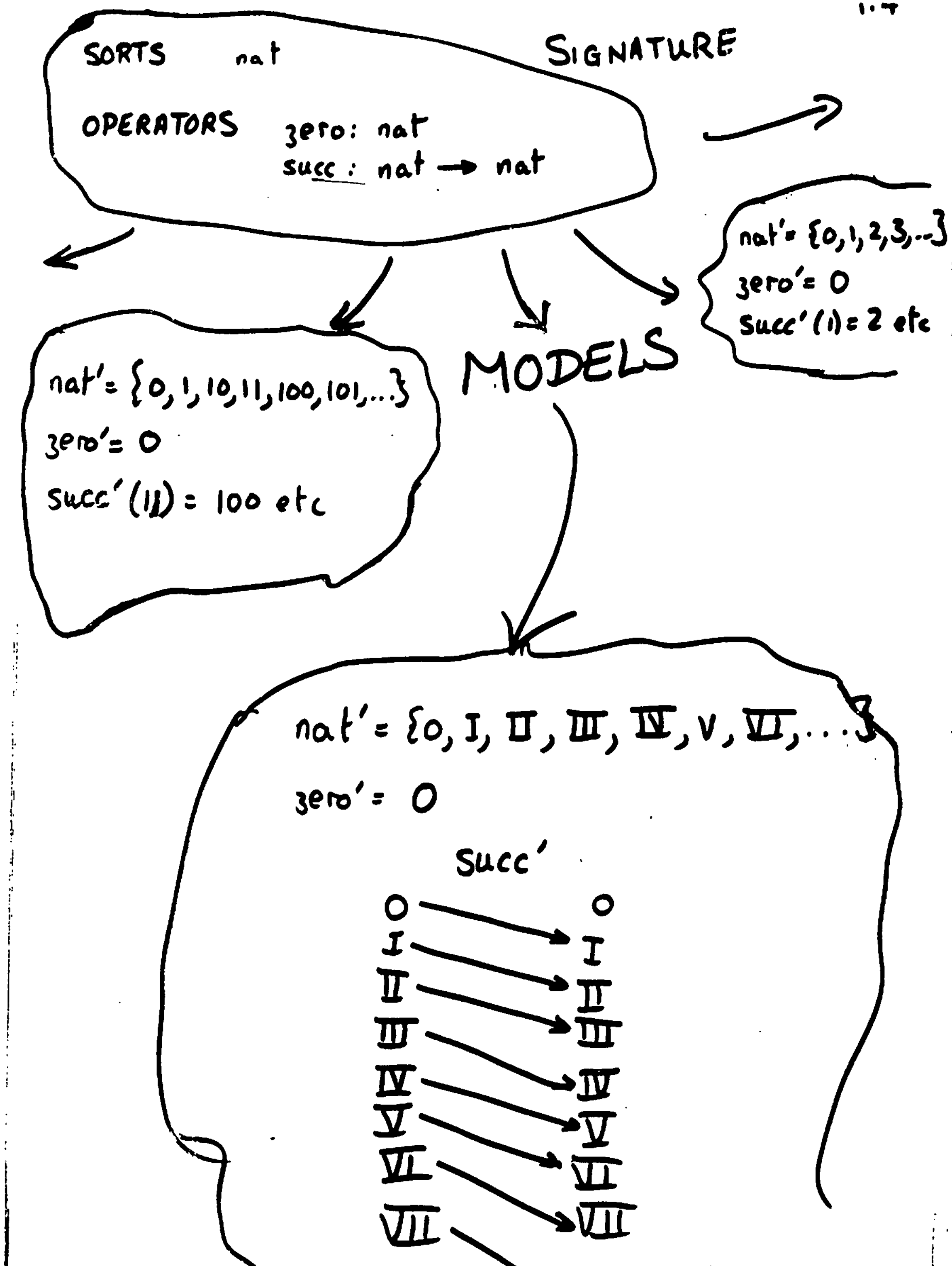
Counting numbers
a zero
a successor function

SORTS nat

OPERATORS zero : nat

succ : nat → nat

SIGNATURE



SORTS $\quad$ *stack* $\quad$ *item*

OPERATORS

a, b, c, d : item

empty : stack

top : stack $\rightarrow$ item

pop : stack $\rightarrow$ stack

push : item stack $\rightarrow$ stack.

Intention

item' = $\{a, b, c, d\}$

stack' = $\left\{ \lfloor\ \rfloor, \left\lfloor a \right\rfloor, \left\lfloor b \right\rfloor, \left\lfloor c \right\rfloor, \left\lfloor \begin{matrix} a \\ b \end{matrix} \right\rfloor, \left\lfloor \begin{matrix} a \\ a \\ b \end{matrix} \right\rfloor, \left\lfloor \begin{matrix} c \\ b \\ c \\ a \end{matrix} \right\rfloor, \text{etc..} \right\}$

mpty' = $\lfloor\ \rfloor$

op' $\left(\left\lfloor \begin{matrix} b \\ c \\ a \end{matrix} \right\rfloor \right) = b$ $\qquad$ pop' $\left(\left\lfloor \begin{matrix} a \\ b \\ c \\ a \end{matrix} \right\rfloor \right) = \left\lfloor \begin{matrix} b \\ c \\ a \end{matrix} \right\rfloor$

ush' $\left(a, \left\lfloor \begin{matrix} b \\ c \\ a \end{matrix} \right\rfloor \right) = \left\lfloor \begin{matrix} a \\ b \\ c \\ a \end{matrix} \right\rfloor$

etc

pop' (push' (a, $\lfloor \blacksquare \rfloor$))

= pop' ($\left\lfloor \begin{matrix} a \\ \blacksquare \end{matrix} \right\rfloor$)

= $\lfloor \blacksquare \rfloor$

We are only interested in models which.....

SATISFY THE EQUATION

pop (push (i, s)) = s

" for any item i and stack s "

We restrict the class of models by augmenting our signature by **equations**

eg

SORTS stack item
OPERATORS a, b, c, d : item
error-item : item
empty, underflow : stack
top : stack → item
pop : stack → stack
push : item.stack → stack
EQUATIONS
i: item s: stack
top(push(i,s)) = i
pop(push(i,s)) = s
pop(empty) = underflow
⋮

PRESENTATION of a THEORY

↙

ALGEBRAS over the THEORY

SORTS nat
OPERATORS zero : nat
succ : nat → nat

nat' = {0}
zero' = 0
succ'(0) = 0

! ! ! ! !

Intuition Demands

nat' = { zero, succ(zero), succ(succ(zero)), ... }
"are all different"

zero' = zero

succ'

zero → zero
succ(zero) → succ(zero)
succ(succ(zero)) → succ(succ(zero))
⋮ succ(succ(succ(zero)))

TERM ALGEBRA

NOTICE

0	0
succ′(0) = 1	succ′(0) = I
succ′(succ′(0)) = 2	succ′(succ′(0)) = II
…cc′(succ′(succ′(0))) = 3	succ′(succ′(succ′(0))) = III
⋮	⋮

Both are essentially

RELABELLED versions of the

TERM ALGEBRA

Terms

items

a, b, c, d, top(empty), top(push(a, empty))....

stacks

empty, push(a, empty), push(top(empty), empty)

......

EQUATIONS generate an

EQUIVALENCE

HOW??

(Congruence)

every "item-term" is 'equivalent' to precisely one of....

error-item, a, b, c, d

and every "stack-term" is equivalent to precisely one of...

underflow, empty, push(a, empty), push(b, empty)

push(c, empty) ..... push(a, push(b, empty))

⋮

.... push(letter, earlier term)

INITIAL ALGEBRA

Write out the distinct terms of sort nat_3 in an initial algebra of.....

SORTS nat_3

OPERATORS $zero : nat_3$

$succ : nat_3 \rightarrow nat_3$

EQUATIONS

$i : nat_3$

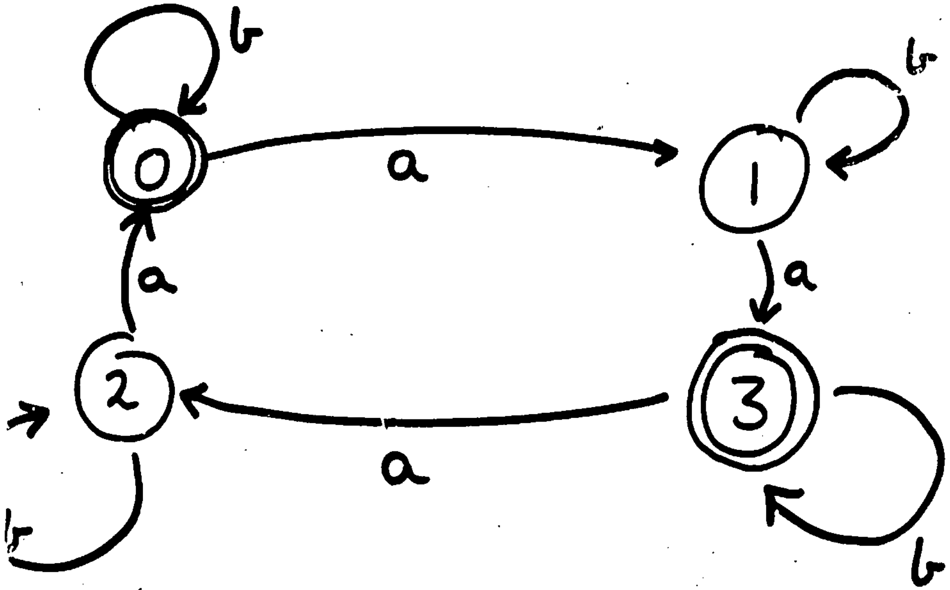
$(succ(succ(succ(i)))) = i$

In both Stack & Nat we were interested only in "models" (algebras) which looked like the Term algebra (initial algebra) Initial Interpretation.

Ofte we are interested in all models (or algebras) of a signature (or theory)

LOOSE INTERPRETATION

FINITE STATE AUTOMATON



States

Alphabet "transitions" "events"

art

NAL

A set of States

A set of Events (Alphabet)

An initial state

A final state

Given an event and a state we have rule next → state

SORTS state , alphabet

OPERATIONS

start : state

final : state

next : alphabet state → state.

SUMMARY

SIGNATURE

SORT, OPERATOR – **SYMBOLS**

MODEL

CARRIER SETS

TERMS

Term Algebra

EQUATIONS

THEORY PRESENTATION

ALGEBRAS

Intitial Algebras

Initial v.s. LOOSE

Interpretations