Data-Stack Theory

syntax

stack	all stacks of items of type X
empty	a stack containing no items
push	a function that takes a stack and an item and gives back another stack
pop	a function that takes a stack and gives back another stack
top	a function that takes a stack and gives back an item

Data-Stack Theory

syntax

→	*stack*	all stacks of items of type X
	empty	a stack containing no items
	push	a function that takes a stack and an item and gives back another stack
	pop	a function that takes a stack and gives back another stack
	top	a function that takes a stack and gives back an item

Data-Stack Theory

syntax

	stack	all stacks of items of type X
→	*empty*	a stack containing no items
	push	a function that takes a stack and an item and gives back another stack
	pop	a function that takes a stack and gives back another stack
	top	a function that takes a stack and gives back an item

Data-Stack Theory

syntax

	stack	all stacks of items of type X
	empty	a stack containing no items
→	*push*	a function that takes a stack and an item and gives back another stack
	pop	a function that takes a stack and gives back another stack
	top	a function that takes a stack and gives back an item

Data-Stack Theory

syntax

	stack	all stacks of items of type X
	empty	a stack containing no items
	push	a function that takes a stack and an item and gives back another stack
→	*pop*	a function that takes a stack and gives back another stack
	top	a function that takes a stack and gives back an item

Data-Stack Theory

syntax

stack	all stacks of items of type X
empty	a stack containing no items
push	a function that takes a stack and an item and gives back another stack
pop	a function that takes a stack and gives back another stack
→ *top*	a function that takes a stack and gives back an item

Data-Stack Theory

axioms

Data-Stack Theory

axioms

empty: *stack*

push: $stack \to X \to stack$

pop: $stack \to stack$

top: $stack \to X$

Data-Stack Theory

axioms

empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*

empty

Data-Stack Theory

axioms

$empty: stack$

$push: stack \rightarrow X \rightarrow stack$

$pop: stack \rightarrow stack$

$top: stack \rightarrow X$

$$empty \rightarrow s1$$

Data-Stack Theory

axioms

$empty$: $stack$

$push$: $stack \rightarrow X \rightarrow stack$

pop: $stack \rightarrow stack$

top: $stack \rightarrow X$

$$empty \rightarrow s1 \rightarrow s2$$

Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$pop: stack \to stack$

$top: stack \to X$

$$empty \to s1 \to s2 \to s3$$

Data-Stack Theory

axioms

empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*

empty→*s1*→*s2*→*s3*→*s4*

Data-Stack Theory

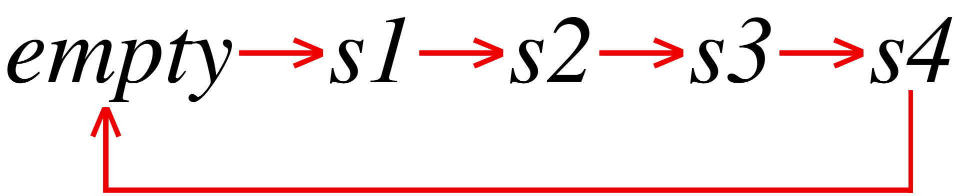
axioms

empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*



Data-Stack Theory

axioms

$empty$: $stack$

$push$: $stack \rightarrow X \rightarrow stack$

pop: $stack \rightarrow stack$

top: $stack \rightarrow X$

$push\ s\ x \neq empty$

$$empty \rightarrow s1 \rightarrow s2 \rightarrow s3 \rightarrow s4$$

Data-Stack Theory

axioms

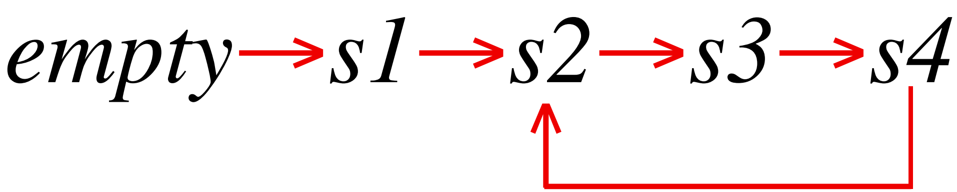
empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*

push s x ≠ *empty*



Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$pop: stack \to stack$

$top: stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \ \ = \ \ s=t \land x=y$

$empty \to s1 \to s2 \to s3 \to s4 \to \ldots\ldots\ldots$

Data-Stack Theory

axioms

empty: *stack*

push: $stack \to X \to stack$

pop: $stack \to stack$

top: $stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \wedge x{=}y$

$empty \to s1 \to s2 \to s3 \to s4 \to \ldots\ldots\ldots$

$\ldots\ldots\ldots \to t \to u \to v \to w \to \ldots\ldots\ldots$

Data-Stack Theory

axioms

→ $empty: stack$

→ $push: stack \rightarrow X \rightarrow stack$

$pop: stack \rightarrow stack$

$top: stack \rightarrow X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \wedge x{=}y$

→ $empty, push\ stack\ X: stack$

$empty \rightarrow s1 \rightarrow s2 \rightarrow s3 \rightarrow s4 \rightarrow ....$

$.......... \rightarrow t \rightarrow u \rightarrow v \rightarrow w \rightarrow ....$

Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$pop: stack \to stack$

$top: stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$\rightarrow$ $empty, push\ stack\ X: stack$

$\rightarrow$ $empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$$empty \to s1 \to s2 \to s3 \to s4 \to \ldots\ldots\ldots$$

Data-Stack Theory

axioms

$empty: stack$

$push: stack \rightarrow X \rightarrow stack$

$pop: stack \rightarrow stack$

$top: stack \rightarrow X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$pop: stack \to stack$

$top: stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$pop: stack \to stack$

$top: stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s=t \land x=y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$pop: stack \to stack$

$top: stack \to X$

$push\ s\ x \ne empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

Data-Stack Theory

axioms

$empty\colon stack$

$push\colon stack \to X \to stack$

$pop\colon stack \to stack$

$top\colon stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty,\ push\ stack\ X\colon stack$

$empty,\ push\ B\ X\colon B \quad \Rightarrow \quad stack\colon B$

$P\ empty \land \forall s\colon stack\cdot\ \forall x\colon X\cdot\ P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s\colon stack\cdot\ P\ s$

↑

Data-Stack Theory

axioms

$empty: stack$

$push: stack \rightarrow X \rightarrow stack$

$pop: stack \rightarrow stack$

$top: stack \rightarrow X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$pop: stack \to stack$

$top: stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

Data-Stack Theory

axioms

$empty: stack$

$push: stack \rightarrow X \rightarrow stack$

$pop: stack \rightarrow stack$

$top: stack \rightarrow X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Data-Stack Theory

implementation

Data-Stack Theory

implementation

stack

empty

push

pop

top

Data-Stack Theory

implementation

$$\begin{aligned} stack &= \\ empty &= \\ push &= \\ pop &= \\ top &= \end{aligned}$$

Data-Stack Theory

implementation

$$stack \quad = \quad [*int]$$

$$empty \quad =$$

$$push \quad =$$

$$pop \quad =$$

$$top \quad =$$

Data-Stack Theory

implementation

$$
\begin{aligned}
stack &= [*int] \\
empty &= [nil] \\
push &= \\
pop &= \\
top &=
\end{aligned}
$$

Data-Stack Theory

implementation

$stack$	$=$	$[*int]$
$empty$	$=$	$[nil]$
$push$	$=$	$\langle s\text{: } stack\cdot \langle x\text{: } int\cdot s;;[x]\rangle\rangle$
pop	$=$	
top	$=$	

Data-Stack Theory

implementation

$stack \quad = \quad [*int]$

$empty \quad = \quad [nil]$

$push \quad = \quad \langle s\text{: } stack\cdot\ \langle x\text{: } int\cdot\ s;;[x]\rangle\rangle$

$pop \quad = \quad \langle s\text{: } stack\cdot\ \textbf{if } s{=}empty \textbf{ then } empty \textbf{ else } s\ [0;..\#s{-}1]\ \textbf{fi}\rangle$

$top \quad =$

Data-Stack Theory

implementation

$stack = [*int]$

$empty = [nil]$

$push = \langle s: stack\cdot \langle x: int\cdot s;;[x]\rangle\rangle$

$pop = \langle s: stack\cdot \textbf{if } s{=}empty \textbf{ then } empty \textbf{ else } s\ [0;..\#s{-}1] \textbf{ fi}\rangle$

$top = \langle s: stack\cdot \textbf{if } s{=}empty \textbf{ then } 0 \textbf{ else } s\ (\#s{-}1) \textbf{ fi}\rangle$

Data-Stack Theory

proof

Data-Stack Theory

proof

Prove that the axioms of the theory are satisfied by the definitions of the implementation.

Data-Stack Theory

proof

Prove that the axioms of the theory are satisfied by the definitions of the implementation.

(the axioms of the theory) $\Leftarrow$ (the definitions of the implementation)

Data-Stack Theory

proof

Prove that the axioms of the theory are satisfied by the definitions of the implementation.

(the axioms of the theory) $\Leftarrow$ (the definitions of the implementation)

specification $\Leftarrow$ implementation

Data-Stack Theory

proof (last axiom):

	$top\ (push\ s\ x) = x$	definition of *push*
=	$top\ (\langle s{:}\ stack\cdot\ \langle x{:}\ int\cdot\ s;;[x]\rangle\rangle\ s\ x) = x$	apply function
=	$top\ (s;;[x]) = x$	definition of *top*
=	$\langle s{:}\ stack\cdot\ \textbf{if}\ s{=}empty\ \textbf{then}\ 0\ \textbf{else}\ s\ (\#s{-}1)\ \textbf{fi}\rangle\ (s;;[x]) = x$	apply function
=	$\textbf{if}\ s;;[x]{=}empty\ \textbf{then}\ 0\ \textbf{else}\ (s;;[x])\ (\#(s;;[x]){-}1)\ \textbf{fi} = x$	definition of *empty*
=	$\textbf{if}\ s;;[x]{=}[nil]\ \textbf{then}\ 0\ \textbf{else}\ (s;;[x])\ (\#(s;;[x]){-}1)\ \textbf{fi} = x$	simplify the **if** and the index
=	$(s;;[x])\ (\#s) = x$	index the list
=	$x = x$	reflexive law
=	$\top$	

Data-Stack Theory

proof (last axiom):

	$top\ (push\ s\ x) = x$	→ definition of *push*
=	$top\ (\langle s{:}\ stack\cdot \langle x{:}\ int\cdot s{;;}[x]\rangle\rangle\ s\ x) = x$	apply function
=	$top\ (s{;;}[x]) = x$	→ definition of *top*
=	$\langle s{:}\ stack\cdot \textbf{if}\ s{=}empty\ \textbf{then}\ 0\ \textbf{else}\ s\ (\#s{-}1)\ \textbf{fi}\rangle\ (s{;;}[x]) = x$	apply function
=	$\textbf{if}\ s{;;}[x]{=}empty\ \textbf{then}\ 0\ \textbf{else}\ (s{;;}[x])\ (\#(s{;;}[x]){-}1)\ \textbf{fi} = x$	→ definition of *empty*
=	$\textbf{if}\ s{;;}[x]{=}[nil]\ \textbf{then}\ 0\ \textbf{else}\ (s{;;}[x])\ (\#(s{;;}[x]){-}1)\ \textbf{fi} = x$	simplify the **if** and the index
=	$(s{;;}[x])\ (\#s) = x$	index the list
=	$x = x$	reflexive law
=	$\top$	

Data-Stack Theory

proof (last axiom):

	$top\ (push\ s\ x) = x$	definition of *push*
=	$top\ (\langle s{:}\ stack\cdot \langle x{:}\ int\cdot s;;[x]\rangle\rangle\ s\ x) = x$	→ apply function
=	$top\ (s;;[x]) = x$	definition of *top*
=	$\langle s{:}\ stack\cdot \textbf{if}\ s{=}empty\ \textbf{then}\ 0\ \textbf{else}\ s\ (\#s{-}1)\ \textbf{fi}\rangle\ (s;;[x]) = x$	→ apply function
=	$\textbf{if}\ s;;[x]{=}empty\ \textbf{then}\ 0\ \textbf{else}\ (s;;[x])\ (\#(s;;[x]){-}1)\ \textbf{fi} = x$	definition of *empty*
=	$\textbf{if}\ s;;[x]{=}[nil]\ \textbf{then}\ 0\ \textbf{else}\ (s;;[x])\ (\#(s;;[x]){-}1)\ \textbf{fi} = x$	simplify the **if** and the index
=	$(s;;[x])\ (\#s) = x$	→ index the list
=	$x = x$	reflexive law
=	$\top$	

Data-Stack Theory

proof (last axiom):

	$top\ (push\ s\ x) = x$	definition of *push*
$=$	$top\ (\langle s{:}\ stack\cdot \langle x{:}\ int\cdot s;;[x]\rangle\rangle\ s\ x) = x$	apply function
$=$	$top\ (s;;[x]) = x$	definition of *top*
$=$	$\langle s{:}\ stack\cdot \textbf{if } s{=}empty \textbf{ then } 0 \textbf{ else } s\ (\#s{-}1)\ \textbf{fi}\rangle\ (s;;[x]) = x$	apply function
$=$	$\textbf{if } s;;[x]{=}empty \textbf{ then } 0 \textbf{ else } (s;;[x])\ (\#(s;;[x]){-}1)\ \textbf{fi} = x$	definition of *empty*
$=$ $\longrightarrow$	$\textbf{if } s;;[x]{=}[nil] \textbf{ then } 0 \textbf{ else } (s;;[x])\ (\#(s;;[x]){-}1)\ \textbf{fi} = x$	simplify the **if** and the index
$=$	$(s;;[x])\ (\#s) = x$	index the list
$=$	$x = x$	reflexive law
$=$	$\top$	

Data-Stack Theory

usage

var a, b: *stack*

Data-Stack Theory

usage

var a, b: *stack*

a:= *empty*

Data-Stack Theory

usage

var a, b: *stack*

$a:= empty.\ \ b:= push\ a\ 2$

Data-Stack Theory

usage

var a, b: *stack*

$a := empty$. $b := push\ a\ 2$

consistent?

Data-Stack Theory

usage

var a, b: *stack*

$a := empty. \;\; b := push\ a\ 2$

consistent?

yes, we implemented it.

Data-Stack Theory

usage

var a, b: *stack*

$a := empty$. $b := push\ a\ 2$

consistent?

yes, we implemented it.

complete?

Data-Stack Theory

usage

var a, b: *stack*

$a{:=}\ empty.\ \ b{:=}\ push\ a\ 2$

consistent?

yes, we implemented it.

complete?

no, the binary expressions

$pop\ empty = empty$

$top\ empty = 0$

are unclassified.

Data-Stack Theory

usage

var a, b: *stack*

$a{:=}\ empty.\ \ b{:=}\ push\ a\ 2$

consistent?

yes, we implemented it.

complete?

no, the binary expressions

$pop\ empty = empty$

$top\ empty = 0$

are unclassified. Proof: implement twice.

Theory as Firewall

user ensures that		implementer ensures that
only stack properties	theory	**all** stack properties
are relied upon		are provided

Simple Data-Stack Theory

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$pop: stack \to stack$

$top: stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \rightarrow X \rightarrow stack$

→ $pop: stack \rightarrow stack$

$top: stack \rightarrow X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$\rightarrow$ $pop: stack \to stack \quad \Rightarrow \quad pop\ empty: stack$

$top: stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \wedge x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \wedge \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

$\to$ $pop: stack \to stack \quad \Rightarrow \quad pop\ empty: stack$

$top: stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \;=\; s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \;\Rightarrow\; stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \;=\; \forall s: stack \cdot P\ s$

$\to$ $pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

~~$pop: stack \to stack$~~

$top: stack \to X$

$push\ s\ x \ne empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

empty: *stack*

push: *stack*→*X*→*stack*

~~*pop*: *stack*→*stack*~~

→ *top*: *stack*→*X* ⇒ *top empty*: *X*

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \ = \ s{=}t \land x{=}y$

empty, *push stack X*: *stack*

empty, *push B X*: *B* ⇒ *stack*: *B*

$P\ empty \land \forall s\colon stack\cdot \forall x\colon X\cdot P\ s \Rightarrow P(push\ s\ x) \ = \ \forall s\colon stack\cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

~~$pop: stack \to stack$~~

→ $top: stack \to X \quad \Rightarrow \quad top\ empty: X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

→ $top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

~~$pop: stack \to stack$~~

~~$top: stack \to X$~~

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$P\ empty \land \forall s: stack\cdot \forall x: X\cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack\cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

~~$pop: stack \to stack$~~

~~$top: stack \to X$~~

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

$\longrightarrow$ $P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

empty: *stack*

push: *stack*→*X*→*stack*

~~*pop*: *stack*→*stack*~~

~~*top*: *stack*→*X*~~

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

→ $empty, push\ stack\ X: stack$

→ $empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

→ $P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

~~$pop: stack \to stack$~~

~~$top: stack \to X$~~

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

~~$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$~~

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \rightarrow X \rightarrow stack$

~~$pop: stack \rightarrow stack$~~

~~$top: stack \rightarrow X$~~

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \wedge x{=}y$

$empty, push\ stack\ X: stack$

→ $empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$

~~$P\ empty \wedge \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$~~

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

$empty$: $stack$

$push$: $stack \to X \to stack$

~~pop: $stack \to stack$~~

~~top: $stack \to X$~~

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s=t \wedge x=y$

$empty, push\ stack\ X$: $stack$

~~$empty, push\ B\ X$: $B \quad \Rightarrow \quad stack$: B~~

~~$P\ empty \wedge \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$~~

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

→ $empty$: $stack$

$push$: $stack \to X \to stack$

~~pop: $stack \to stack$~~

~~top: $stack \to X$~~

→ $push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$

→ $empty, push\ stack\ X$: $stack$

~~$empty, push\ B\ X$: $B \quad \Rightarrow \quad stack$: B~~

~~$P\ empty \land \forall s{:}\ stack \cdot \forall x{:}\ X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s{:}\ stack \cdot P\ s$~~

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

~~*empty*: *stack*~~ $\qquad$ *stack* ‡ *null*

push: *stack*→*X*→*stack*

~~*pop*: *stack*→*stack*~~

~~*top*: *stack*→*X*~~

~~*push s x* ‡ *empty*~~

push s x = *push t y* $\;=\;$ *s*=*t* ∧ *x*=*y*

~~*empty*, *push stack X*: *stack*~~

~~*empty*, *push B X*: *B* ⇒ *stack*: *B*~~

~~*P empty* ∧ ∀*s*: *stack*· ∀*x*: *X*· *P s* ⇒ *P*(*push s x*) = ∀*s*: *stack*· *P s*~~

pop (*push s x*) = *s*

top (*push s x*) = *x*

Simple Data-Stack Theory

axioms

~~$empty: stack$~~ $stack \neq null$

$push: stack \rightarrow X \rightarrow stack$

~~$pop: stack \rightarrow stack$~~

~~$top: stack \rightarrow X$~~

~~$push\ s\ x \neq empty$~~

→ $push\ s\ x = push\ t\ y \quad = \quad s{=}t \wedge x{=}y$

~~$empty, push\ stack\ X: stack$~~

~~$empty, push\ B\ X: B \quad \Rightarrow \quad stack: B$~~

~~$P\ empty \wedge \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s: stack \cdot P\ s$~~

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

~~*empty*: *stack*~~ $\qquad$ *stack* ≠ *null*

push: *stack*→*X*→*stack*

~~*pop*: *stack*→*stack*~~

~~*top*: *stack*→*X*~~

~~*push s x* ≠ *empty*~~

~~*push s x* = *push t y* $=$ *s*=*t* ∧ *x*=*y*~~

~~*empty*, *push stack X*: *stack*~~

~~*empty*, *push B X*: *B* ⇒ *stack*: *B*~~

~~*P empty* ∧ ∀*s*: *stack*· ∀*x*: *X*· *P s* ⇒ *P*(*push s x*) $=$ ∀*s*: *stack*· *P s*~~

pop (*push s x*) = *s*

top (*push s x*) = *x*

Simple Data-Stack Theory

axioms

→ ~~*empty*: *stack*~~ *stack* ≠ *null*

push: *stack*→*X*→*stack*

~~*pop*: *stack*→*stack*~~

~~*top*: *stack*→*X*~~

~~*push s x* ≠ *empty*~~

~~*push s x* = *push t y* ≡ *s*=*t* ∧ *x*=*y*~~

~~*empty*, *push stack X*: *stack*~~

~~*empty*, *push B X*: *B* ⇒ *stack*: *B*~~

~~*P empty* ∧ ∀*s*: *stack*· ∀*x*: *X*· *P s* ⇒ *P*(*push s x*) ≡ ∀*s*: *stack*· *P s*~~

pop (*push s x*) = *s*

top (*push s x*) = *x*

Simple Data-Stack Theory

axioms

$empty: stack$ $\qquad stack \neq null$

→ $push: stack \to X \to stack$

$pop: stack \to stack$

$top: stack \to X$

$push\ s\ x \neq empty$

$push\ s\ x = push\ t\ y \;\;=\;\; s=t \land x=y$

$empty, push\ stack\ X: stack$

$empty, push\ B\ X: B \;\;\Rightarrow\;\; stack: B$

$P\ empty \land \forall s: stack \cdot \forall x: X \cdot P\ s \Rightarrow P(push\ s\ x) \;\;=\;\; \forall s: stack \cdot P\ s$

$pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

~~$empty$: $stack$~~ $\qquad$ $stack \neq null$

$push$: $stack \rightarrow X \rightarrow stack$

~~pop: $stack \rightarrow stack$~~

~~top: $stack \rightarrow X$~~

~~$push\ s\ x \neq empty$~~

~~$push\ s\ x = push\ t\ y \quad = \quad s{=}t \land x{=}y$~~

~~$empty, push\ stack\ X$: $stack$~~

~~$empty, push\ B\ X$: $B \quad \Rightarrow \quad stack$: B~~

~~$P\ empty \land \forall s$: $stack \cdot \forall x$: $X \cdot P\ s \Rightarrow P(push\ s\ x) \quad = \quad \forall s$: $stack \cdot P\ s$~~

→ $pop\ (push\ s\ x) = s$

$top\ (push\ s\ x) = x$

Simple Data-Stack Theory

axioms

~~*empty*: *stack*~~ $\qquad$ *stack* ≠ *null*

push: *stack*→*X*→*stack*

~~*pop*: *stack*→*stack*~~

~~*top*: *stack*→*X*~~

~~*push s x* ≠ *empty*~~

~~*push s x* = *push t y* $\;=\;$ *s*=*t* ∧ *x*=*y*~~

~~*empty*, *push stack X*: *stack*~~

~~*empty*, *push B X*: *B* $\;\Rightarrow\;$ *stack*: *B*~~

~~*P empty* ∧ ∀*s*: *stack*· ∀*x*: *X*· *P s* ⇒ *P*(*push s x*) $\;=\;$ ∀*s*: *stack*· *P s*~~

pop (*push s x*) = *s*

⟶ $\qquad$ *top* (*push s x*) = *x*

Data-Queue Theory

Data-Queue Theory

emptyq: *queue*

Data-Queue Theory

emptyq: *queue*

join: $queue \rightarrow X \rightarrow queue$

Data-Queue Theory

emptyq: *queue*

join q x: *queue*

Data-Queue Theory

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \wedge x{=}y$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \wedge x{=}y$

$leave: queue \rightarrow queue$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \wedge x{=}y$

$leave\ q: queue$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q=r \land x=y$

$q \neq emptyq \;\Rightarrow\; leave\ q: queue$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \wedge x{=}y$

$q{\neq}emptyq \ \Rightarrow\ leave\ q: queue$

$front: queue{\rightarrow}X$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \land x{=}y$

$q{\neq}emptyq \ \Rightarrow\ leave\ q: queue$

$front\ q: X$

Data-Queue Theory

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \land x{=}y$

$q{\neq}emptyq \ \Rightarrow\ leave\ q$: $queue$

$q{\neq}emptyq \ \Rightarrow\ front\ q$: X

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \land x{=}y$

$q{\neq}emptyq \ \Rightarrow\ leave\ q: queue$

$q{\neq}emptyq \ \Rightarrow\ front\ q: X$

$emptyq, join\ B\ X: B \quad \Rightarrow \quad queue: B$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \land x{=}y$

$q{\neq}emptyq \ \Rightarrow\ leave\ q: queue$

$q{\neq}emptyq \ \Rightarrow\ front\ q: X$

$emptyq, join\ B\ X: B \quad \Rightarrow \quad queue: B$

$leave\ (join\ emptyq\ x) = emptyq$

$q{\neq}emptyq \ \Rightarrow\ leave\ (join\ q\ x) = join\ (leave\ q)\ x)$

$front\ (join\ emptyq\ x) = x$

$q{\neq}emptyq \ \Rightarrow\ front\ (join\ q\ x) = front\ \ q$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \wedge x{=}y$

$q{\neq}emptyq \;\Rightarrow\; leave\ q: queue$

$q{\neq}emptyq \;\Rightarrow\; front\ q: X$

$emptyq, join\ B\ X: B \;\Rightarrow\; queue: B$

$leave\ (join\ emptyq\ x) = emptyq$

$q{\neq}emptyq \;\Rightarrow\; leave\ (join\ q\ x) = join\ (leave\ q)\ x)$

$front\ (join\ emptyq\ x) = x$

$q{\neq}emptyq \;\Rightarrow\; front\ (join\ q\ x) = front\ \ q$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \ \ = \ \ q{=}r \wedge x{=}y$

$q{\neq}emptyq \ \Rightarrow\ leave\ q: queue$

$q{\neq}emptyq \ \Rightarrow\ front\ q: X$

$emptyq, join\ B\ X: B \ \ \Rightarrow\ \ queue: B$

$leave\ (join\ emptyq\ x) = emptyq$

→ $q{\neq}emptyq \ \Rightarrow\ leave\ (join\ q\ x) = join\ (leave\ q)\ x)$

$front\ (join\ emptyq\ x) = x$

$q{\neq}emptyq \ \Rightarrow\ front\ (join\ q\ x) = front\ \ q$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \neq emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \wedge x{=}y$

$q{\neq}emptyq \;\Rightarrow\; leave\ q: queue$

$q{\neq}emptyq \;\Rightarrow\; front\ q: X$

$emptyq, join\ B\ X: B \;\Rightarrow\; queue: B$

$leave\ (join\ emptyq\ x) = emptyq$

$q{\neq}emptyq \;\Rightarrow\; leave\ (join\ q\ x) = join\ (leave\ q)\ x)$

→ $front\ (join\ emptyq\ x) = x$

$q{\neq}emptyq \;\Rightarrow\; front\ (join\ q\ x) = front\ \ q$

Data-Queue Theory

$emptyq: queue$

$join\ q\ x: queue$

$join\ q\ x \ne emptyq$

$join\ q\ x = join\ r\ y \quad = \quad q{=}r \wedge x{=}y$

$q{\ne}emptyq \ \Rightarrow\ leave\ q: queue$

$q{\ne}emptyq \ \Rightarrow\ front\ q: X$

$emptyq, join\ B\ X: B \quad \Rightarrow \quad queue: B$

$leave\ (join\ emptyq\ x) = emptyq$

$q{\ne}emptyq \ \Rightarrow\ leave\ (join\ q\ x) = join\ (leave\ q)\ x)$

$front\ (join\ emptyq\ x) = x$

$q{\ne}emptyq \ \Rightarrow\ front\ (join\ q\ x) = front\ \ q$

Strong Data-Tree Theory

Strong Data-Tree Theory

emptree: *tree*

Strong Data-Tree Theory

emptree: *tree*

graft: $tree \to X \to tree \to tree$

Strong Data-Tree Theory

$emptree: tree$

$graft: tree \to X \to tree \to tree$

$emptree, graft\ B\ X\ B: B \ \Rightarrow\ tree: B$

Strong Data-Tree Theory

$emptree\colon tree$

$graft\colon tree \to X \to tree \to tree$

$emptree, graft\ B\ X\ B\colon B \ \Rightarrow\ tree\colon B$

$graft\ t\ x\ u \neq emptree$

Strong Data-Tree Theory

$emptree\colon tree$

$graft\colon tree \to X \to tree \to tree$

$emptree, graft\ B\ X\ B\colon B \;\Rightarrow\; tree\colon B$

$graft\ t\ x\ u \neq emptree$

$graft\ t\ x\ u = graft\ v\ y\ w \;\;=\;\; t{=}v \wedge x{=}y \wedge u{=}w$

Strong Data-Tree Theory

$emptree \colon tree$

$graft \colon tree \rightarrow X \rightarrow tree \rightarrow tree$

$emptree,\ graft\ B\ X\ B \colon B \ \Rightarrow\ tree \colon B$

$graft\ t\ x\ u \neq emptree$

$graft\ t\ x\ u = graft\ v\ y\ w \ \ = \ \ t{=}v \land x{=}y \land u{=}w$

$left\ (graft\ t\ x\ u) = t$

$root\ (graft\ t\ x\ u) = x$

$right\ (graft\ t\ x\ u) = u$

Weak Data-Tree Theory

Weak Data-Tree Theory

$tree \neq null$

$graft\ t\ x\ u: tree$

$left\ (graft\ t\ x\ u) = t$

$root\ (graft\ t\ x\ u) = x$

$right\ (graft\ t\ x\ u) = u$

Data-Tree Implementation

Data-Tree Implementation

$tree \;=\; emptree, graft\ tree\ int\ tree$

$emptree \;=\; [nil]$

$graft \;=\; \langle t: tree\cdot \langle x: int\cdot \langle u: tree\cdot [t; x; u]\rangle\rangle\rangle$

$left \;=\; \langle t: tree\cdot t\ 0\rangle$

$right \;=\; \langle t: tree\cdot t\ 2\rangle$

$root \;=\; \langle t: tree\cdot t\ 1\rangle$

Data-Tree Implementation

$tree \;=\; emptree, graft\; tree\; int\; tree$

→ $emptree \;=\; [nil]$

$graft \;=\; \langle t{:}\ tree\cdot \langle x{:}\ int\cdot \langle u{:}\ tree\cdot [t;\ x;\ u]\rangle\rangle\rangle$

$left \;=\; \langle t{:}\ tree\cdot t\ 0\rangle$

$right \;=\; \langle t{:}\ tree\cdot t\ 2\rangle$

$root \;=\; \langle t{:}\ tree\cdot t\ 1\rangle$

Data-Tree Implementation

$tree \;=\; emptree, graft\ tree\ int\ tree$

$emptree \;=\; [nil]$

→ $graft \;=\; \langle t\colon tree\cdot \langle x\colon int\cdot \langle u\colon tree\cdot [t; x; u]\rangle\rangle\rangle$

$left \;=\; \langle t\colon tree\cdot t\ 0\rangle$

$right \;=\; \langle t\colon tree\cdot t\ 2\rangle$

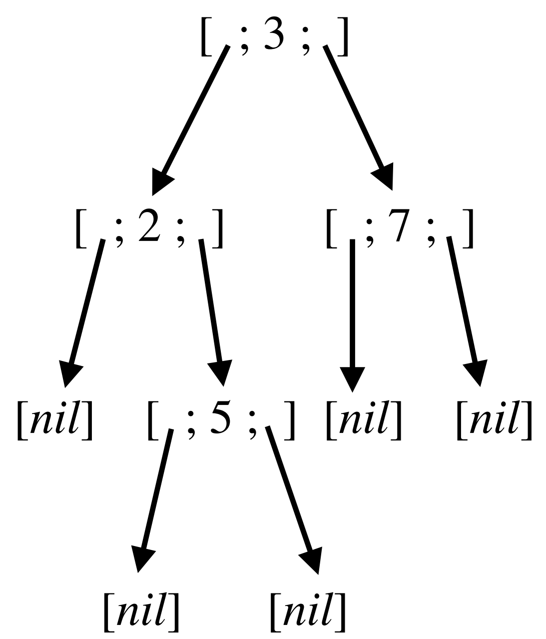
$root \;=\; \langle t\colon tree\cdot t\ 1\rangle$

Data-Tree Implementation

$[[[nil]; 2; [[nil]; 5; [nil]]]; 3; [[nil]; 7; [nil]]]$

Data-Tree Implementation

[[[*nil*]; 2; [[*nil*]; 5; [*nil*]]]; 3; [[*nil*]; 7; [*nil*]]]



Data-Tree Implementation

$tree\ =\ emptree, graft\ tree\ int\ tree$

$emptree\ =\ 0$

$graft\ =\ \langle t\text{: } tree\cdot \langle x\text{: } int\cdot \langle u\text{: } tree\cdot \text{“left”}\to t \mid \text{“root”}\to x \mid \text{“right”}\to u\rangle\rangle\rangle$

$left\ =\ \langle t\text{: } tree\cdot t\ \text{“left”}\rangle$

$right\ =\ \langle t\text{: } tree\cdot t\ \text{“right”}\rangle$

$root\ =\ \langle t\text{: } tree\cdot t\ \text{“root”}\rangle$

Data-Tree Implementation

$tree = emptree, graft\ tree\ int\ tree$

→ $emptree = 0$

$graft = \langle t: tree\cdot \langle x: int\cdot \langle u: tree\cdot \text{“left”}\rightarrow t \mid \text{“root”}\rightarrow x \mid \text{“right”}\rightarrow u\rangle\rangle\rangle$

$left = \langle t: tree\cdot t\ \text{“left”}\rangle$

$right = \langle t: tree\cdot t\ \text{“right”}\rangle$

$root = \langle t: tree\cdot t\ \text{“root”}\rangle$

Data-Tree Implementation

$$tree \;=\; emptree,\, graft\ tree\ int\ tree$$

$$emptree \;=\; 0$$

→ $$graft \;=\; \langle t\text{: } tree\cdot\ \langle x\text{: } int\cdot\ \langle u\text{: } tree\cdot\ \text{“left”}\!\to t \mid \text{“root”}\!\to x \mid \text{“right”}\!\to u\rangle\rangle\rangle$$

$$left \;=\; \langle t\text{: } tree\cdot\ t\ \text{“left”}\rangle$$

$$right \;=\; \langle t\text{: } tree\cdot\ t\ \text{“right”}\rangle$$

$$root \;=\; \langle t\text{: } tree\cdot\ t\ \text{“root”}\rangle$$

Data-Tree Implementation

$$
\begin{aligned}
&\text{“left”} \rightarrow \quad (\text{“left”} \rightarrow 0 \\
&\qquad\qquad\quad \mid \text{“root”} \rightarrow 2 \\
&\qquad\qquad\quad \mid \text{“right”} \rightarrow \quad (\text{“left”} \rightarrow 0 \\
&\qquad\qquad\qquad\qquad\qquad\quad \mid \text{“root”} \rightarrow 5 \\
&\qquad\qquad\qquad\qquad\qquad\quad \mid \text{“right”} \rightarrow 0\ )\ ) \\
&\mid \text{“root”} \rightarrow 3 \\
&\mid \text{“right”} \rightarrow \quad (\text{“left”} \rightarrow 0 \\
&\qquad\qquad\quad \mid \text{“root”} \rightarrow 7 \\
&\qquad\qquad\quad \mid \text{“right”} \rightarrow 0\ )
\end{aligned}
$$