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*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*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*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*

empty→*s1*

Data-Stack Theory

axioms

empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*

$$empty \rightarrow s1 \rightarrow s2$$

Data-Stack Theory

axioms

empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*

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

Data-Stack Theory

axioms

empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*

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

Data-Stack Theory

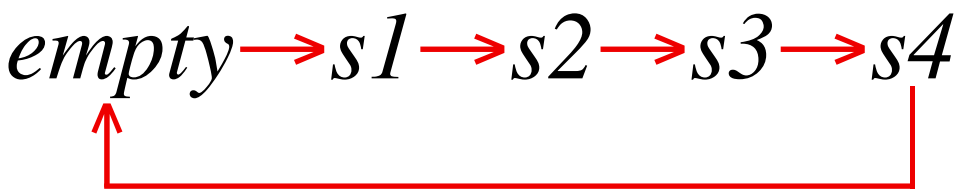
axioms

empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*



Data-Stack Theory

axioms

empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*

push s x ≠ *empty*

empty→*s1*→*s2*→*s3*→*s4*

Data-Stack Theory

axioms

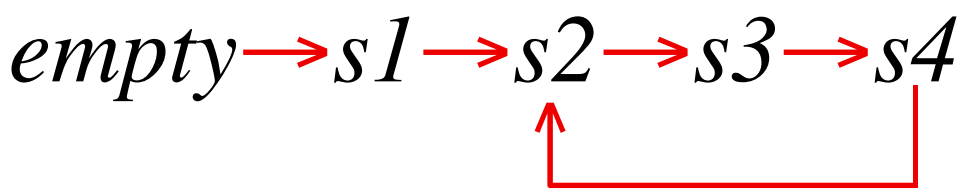
empty: *stack*

push: *stack*→*X*→*stack*

pop: *stack*→*stack*

top: *stack*→*X*

push s x ≠ *empty*



Data-Stack Theory

axioms

$empty$: $stack$

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

pop: $stack \rightarrow stack$

top: $stack \rightarrow X$

$push\ s\ x \neq empty$

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

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

Data-Stack Theory

axioms

$empty$: $stack$

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

pop: $stack \rightarrow stack$

top: $stack \rightarrow X$

$push\ s\ x \neq empty$

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

$$empty \rightarrow s1 \rightarrow s2 \rightarrow s3 \rightarrow s4 \rightarrow \ldots\ldots\ldots$$
$$\ldots\ldots\ldots \rightarrow t \rightarrow u \rightarrow v \rightarrow w \rightarrow \ldots\ldots\ldots$$

Data-Stack Theory

axioms

→ *empty*: *stack*

→ *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→*s1*→*s2*→*s3*→*s4*→.....

.....→*t*→*u*→*v*→*w*→.....

Data-Stack Theory

axioms

$empty : stack$

$push : stack \rightarrow X \rightarrow stack$

$pop : stack \rightarrow stack$

$top : stack \rightarrow X$

$push\ s\ x \neq empty$

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

→ $empty, push\ stack\ X : stack$

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

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

Data-Stack Theory

axioms

$empty$: $stack$

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

pop: $stack \to stack$

top: $stack \to X$

$push\ s\ x \neq empty$

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

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

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

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

Data-Stack Theory

axioms

$$empty: stack$$

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

$$pop: stack \rightarrow stack$$

$$top: stack \rightarrow X$$

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

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

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

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

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

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

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

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

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

Data-Stack Theory

axioms

$$empty : stack$$

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

$$pop : stack \to stack$$

$$top : stack \to X$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Data-Stack Theory

implementation

Data-Stack Theory

implementation

stack

empty

push

pop

top

Data-Stack Theory

implementation

stack =

empty =

push =

pop =

top =

Data-Stack Theory

implementation

$$stack = [*int]$$

$$empty =$$

$$push =$$

$$pop =$$

$$top =$$

Data-Stack Theory

implementation

$$stack = [*int]$$

$$empty = [nil]$$

$$push =$$

$$pop =$$

$$top =$$

Data-Stack Theory

implementation

$$
\begin{array}{lll}
\mathit{stack} & = & [*\mathit{int}] \\
\mathit{empty} & = & [\mathit{nil}] \\
\mathit{push} & = & \langle s\colon \mathit{stack}\cdot \langle x\colon \mathit{int}\cdot\ s;;[x]\rangle\rangle \\
\mathit{pop} & = & \\
\mathit{top} & = &
\end{array}
$$

Data-Stack Theory

implementation

$$
\begin{array}{lll}
\mathit{stack} & = & [*\mathit{int}] \\
\mathit{empty} & = & [\mathit{nil}] \\
\mathit{push} & = & \langle s\colon \mathit{stack}\cdot \langle x\colon \mathit{int}\cdot\ s;;[x]\rangle\rangle \\
\mathit{pop} & = & \langle s\colon \mathit{stack}\cdot\ \textbf{if}\ s{=}\mathit{empty}\ \textbf{then}\ \mathit{empty}\ \textbf{else}\ s\ [0;..\#s{-}1]\ \textbf{fi}\rangle \\
\mathit{top} & = &
\end{array}
$$

Data-Stack Theory

implementation

$$
\begin{array}{lcl}
\mathit{stack} & = & [*\mathit{int}] \\
\mathit{empty} & = & [\mathit{nil}] \\
\mathit{push} & = & \langle s\colon \mathit{stack}\cdot \langle x\colon \mathit{int}\cdot\ s;;[x]\rangle\rangle \\
\mathit{pop} & = & \langle s\colon \mathit{stack}\cdot\ \textbf{if}\ s{=}\mathit{empty}\ \textbf{then}\ \mathit{empty}\ \textbf{else}\ s\ [0;..\#s{-}1]\ \textbf{fi}\rangle \\
\mathit{top} & = & \langle s\colon \mathit{stack}\cdot\ \textbf{if}\ s{=}\mathit{empty}\ \textbf{then}\ 0\ \textbf{else}\ s\ (\#s{-}1)\ \textbf{fi}\rangle
\end{array}
$$

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

Data-Stack Theory

usage

new *a*, *b*: *stack*

Data-Stack Theory

usage

new *a*, *b*: *stack*

a:= *empty*

Data-Stack Theory

usage

new *a*, *b*: *stack*

a:= *empty*. *b*:= *push a* 2

Data-Stack Theory

usage

new *a*, *b*: *stack*

a:= *empty*. *b*:= *push a* 2

consistent?

Data-Stack Theory

usage

new *a*, *b*: *stack*

a:= *empty*. *b*:= *push a* 2

consistent?

yes, we implemented it.

Data-Stack Theory

usage

new *a*, *b*: *stack*

a:= *empty*. *b*:= *push a* 2

consistent?

yes, we implemented it.

complete?

Data-Stack Theory

usage

new *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

new *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: $stack \to X$

$push\ s\ x \neq empty$

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

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

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

$empty$: $stack$

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

→ pop: $stack \rightarrow stack$ $\Rightarrow$ $pop\ empty$: $stack$

top: $stack \rightarrow X$

$push\ s\ x \neq empty$

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

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

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

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

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

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

Simple Data-Stack Theory

axioms

$empty$: $stack$

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

→ pop: $stack \rightarrow stack$ $\Rightarrow$ $pop\ empty$: $stack$

top: $stack \rightarrow X$

$push\ s\ x \neq empty$

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

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

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

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

→ $pop\ (push\ s\ x) = s$

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

Simple Data-Stack Theory

axioms

$empty$: $stack$

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

~~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

$empty: stack$

$push: stack \to X \to stack$

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

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

$push\ s\ x \neq empty$

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

$empty, push\ stack\ X: stack$

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

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

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

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

Simple Data-Stack Theory

axioms

$empty: stack$

$push: stack \to X \to stack$

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

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

$push\ s\ x \neq empty$

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

$empty, push\ stack\ X: stack$

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

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

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

→ $top\ (push\ s\ x) = x$

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: $stack \to X$~~

$push\ s\ x \neq empty$

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

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

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

$push\ s\ x \neq empty$

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

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

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

$empty$: $stack$

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

~~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

$empty$: $stack$

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

~~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: $stack \to X$~~

→ $push\ s\ x \neq empty$

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

→ $empty, push\ stack\ X$: $stack$

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

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

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

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

Simple Data-Stack Theory

axioms

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

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

~~*pop: stack→stack*~~

~~*top: stack→X*~~

~~*push s x ≠ empty*~~

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

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

~~*empty, push B X: B ⇒ 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*~~ $\quad\quad$ $stack \neq null$

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

~~*pop: stack→stack*~~

~~*top: stack→X*~~

~~*push s x ≠ empty*~~

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

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

~~*empty, push B X: B ⇒ 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*~~ $\qquad\qquad$ $stack \neq null$

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

~~*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*~~ $\quad\quad$ $stack \neq null$

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

~~*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→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*~~ $\quad\quad$ $stack \neq null$

$push: stack \rightarrow X \rightarrow stack$

~~*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*~~ $\qquad$ $stack \neq null$

$push: stack \rightarrow X \rightarrow stack$

~~*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*~~

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

→ $top\ (push\ s\ x) = x$

Data-Queue Theory

Data-Queue Theory

emptyq: *queue*

Data-Queue Theory

emptyq: *queue*

join: *queue*→*X*→*queue*

Data-Queue Theory

emptyq: *queue*

join q x: *queue*

Data-Queue Theory

emptyq: *queue*

join q x: *queue*

join q x ≠ *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 ≠ *emptyq*

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

leave: *queue*→*queue*

Data-Queue Theory

emptyq: *queue*

join q x: *queue*

join q x ≠ *emptyq*

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

leave q: *queue*

Data-Queue Theory

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

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

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

Data-Queue Theory

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

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

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

$front$: $queue \rightarrow X$

Data-Queue Theory

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

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

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

$front\ q$: X

Data-Queue Theory

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

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

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

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

Data-Queue Theory

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

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

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

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

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

Data-Queue Theory

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

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

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

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

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

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

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

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

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

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

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

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

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

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

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

$emptyq$: $queue$

$join\ q\ x$: $queue$

$join\ q\ x \neq emptyq$

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

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

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

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

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

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

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

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

Strong Data-Tree Theory

Strong Data-Tree Theory

emptree: *tree*

Strong Data-Tree Theory

emptree: *tree*

graft: *tree*→*X*→*tree*→*tree*

Strong Data-Tree Theory

$emptree$: $tree$

$graft$: $tree \rightarrow X \rightarrow tree \rightarrow tree$

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

Strong Data-Tree Theory

$emptree$: $tree$

$graft$: $tree \rightarrow X \rightarrow tree \rightarrow tree$

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

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

Strong Data-Tree Theory

$emptree$: $tree$

$graft$: $tree \rightarrow X \rightarrow tree \rightarrow tree$

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

$emptree$: $tree$

$graft$: $tree \rightarrow X \rightarrow tree \rightarrow tree$

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

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

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

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

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

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

Weak Data-Tree Theory

Weak Data-Tree Theory

$tree \neq null$

$graft\ t\ x\ u\colon tree$

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

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

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

Data-Tree Implementation

Data-Tree Implementation

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

$$emptree \;=\; [nil]$$

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

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

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

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

Data-Tree Implementation

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

$$\rightarrow \quad emptree \;=\; [nil]$$

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

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

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

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

Data-Tree Implementation

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

$$emptree = [nil]$$

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

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

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

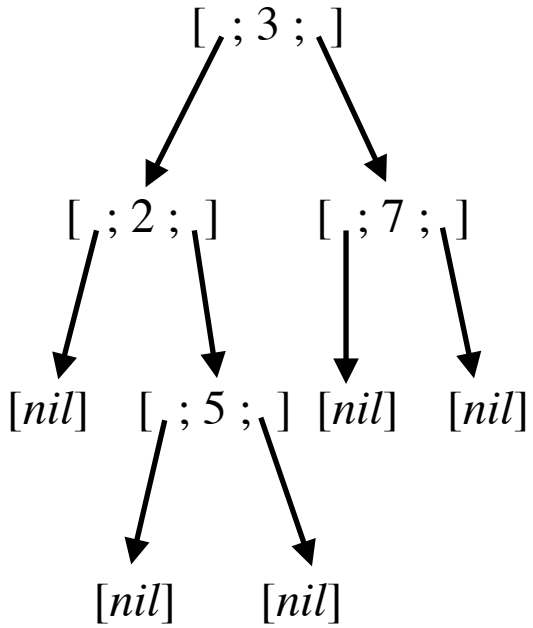
$$root = \langle t{:}\ tree{\cdot}\ t\ 1\rangle$$

Data-Tree Implementation

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

Data-Tree Implementation

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



Data-Tree Implementation

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

$emptree \;=\; 0$

$graft \;=\; \langle t\colon tree\cdot \langle x\colon int\cdot \langle u\colon tree\cdot$ "left"$\rightarrow t \mid$ "root"$\rightarrow x \mid$ "right"$\rightarrow u\rangle\rangle\rangle$

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

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

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

Data-Tree Implementation

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

→ $emptree = 0$

$graft = \langle t\colon tree\cdot \langle x\colon int\cdot \langle u\colon tree\cdot$ "left"$\rightarrow t \mid$ "root"$\rightarrow x \mid$ "right"$\rightarrow u\rangle\rangle\rangle$

$left = \langle t\colon tree\cdot\ t$ "left"$\rangle$

$right = \langle t\colon tree\cdot\ t$ "right"$\rangle$

$root = \langle t\colon tree\cdot\ t$ "root"$\rangle$

Data-Tree Implementation

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

$emptree = 0$

→ $graft = \langle t\colon tree\cdot \langle x\colon int\cdot \langle u\colon tree\cdot$ “left”→$t \mid$ “root”→$x \mid$ “right”→$u\rangle\rangle\rangle$

$left = \langle t\colon tree\cdot\ t$ “left”$\rangle$

$right = \langle t\colon tree\cdot\ t$ “right”$\rangle$

$root = \langle t\colon tree\cdot\ t$ “root”$\rangle$

Data-Tree Implementation

```
"left" →    ("left" → 0
                | "root" → 2
                | "right" →   ("left" → 0
                                  | "root" → 5
                                  | "right" → 0 ) )
| "root" → 3
| "right" →  ("left" → 0
                | "root" → 7
                | "right" → 0 )
```