The Lean standard library

Markus Himmel, Lean FRO

Lean’s “five pillars”

Lean’s “five pillars”

- Formal mathematics

Lean’s “five pillars”

- Formal mathematics
- Software and hardware verification

Lean’s “five pillars”

- Formal mathematics
- Software and hardware verification
- (Verified) software development

Lean’s “five pillars”

- Formal mathematics
- Software and hardware verification
- (Verified) software development
- AI research for mathematics and code synthesis

Lean’s “five pillars”

- Formal mathematics
- Software and hardware verification
- (Verified) software development
- AI research for mathematics and code synthesis
- New math and computer science education methodologies

What’s in the Lean distribution?

What’s in the Lean distribution?

- The language (parser, elaborator, kernel, compiler, runtime, …)

What’s in the Lean distribution?

- The language (parser, elaborator, kernel, compiler, runtime, …)
- The language server

What’s in the Lean distribution?

- The language (parser, elaborator, kernel, compiler, runtime, …)
- The language server
- Tactics (basic like `exact` or `rewrite`, advanced like `grind` or `bv_decide`)

What’s in the Lean distribution?

- The language (parser, elaborator, kernel, compiler, runtime, …)
- The language server
- Tactics (basic like `exact` or `rewrite`, advanced like `grind` or `bv_decide`)
- The metaprogramming framework

What’s in the Lean distribution?

- The language (parser, elaborator, kernel, compiler, runtime, …)
- The language server
- Tactics (basic like `exact` or `rewrite`, advanced like `grind` or `bv_decide`)
- The metaprogramming framework
- **The standard library**

What is the standard library?

What is the standard library?

It’s what makes Lean into a **general-purpose** programming language.

What is the standard library?

It’s what makes Lean into a **general-purpose** programming language.

It’s what makes Lean into a verification platform.

What is the standard library?

It’s what makes Lean into a **general-purpose** programming language.

It’s what makes Lean into a verification platform.

It is a public API.

What is/will be in the standard library?

1. Core types and operations
    a. Basic types
    b. Numeric types, including floating point numbers
    c. Containers
    d. Strings and formatting
2. Language constructs
3. Libraries
4. Operating system abstractions

What is/will be in the standard library?

1. Core types and operations
2. Language constructs
    a. Ranges and iterators
    b. Comparison, ordering, hashing and related type classes
    c. Basic monad infrastructure
3. Libraries
4. Operating system abstractions

What is/will be in the standard library?

1. Core types and operations
2. Language constructs
3. Libraries
    a. Random numbers
    b. Dates and times
4. Operating system abstractions

What is/will be in the standard library?

1. Core types and operations
2. Language constructs
3. Libraries
4. Operating system abstractions
    a. Concurrency and parallelism primitives
    b. Asynchronous I/O
    c. FFI helpers
    d. Environment, file system, processes
    e. Locales

Who is the standard library for?

Who is the standard library for?

All Lean users!

Who is the standard library for?

All Lean users!

- Programmers

Who is the standard library for?

All Lean users!

- Programmers
  - Programmers writing verified software

Who is the standard library for?

All Lean users!

- Programmers
  - Programmers writing verified software
  - Metaprogrammers

Who is the standard library for?

All Lean users!

- Programmers
  - Programmers writing verified software
  - Metaprogrammers
  - Lean developers

Who is the standard library for?

All Lean users!

- Programmers
  - Programmers writing verified software
  - Metaprogrammers
  - Lean developers
  - Library authors

Who is the standard library for?

All Lean users!

- Programmers
  - Programmers writing verified software
  - Metaprogrammers
  - Lean developers
  - Library authors
- Mathlib users

Who is the standard library for?

All Lean users!

- Programmers
  - Programmers writing verified software
  - Metaprogrammers
  - Lean developers
  - Library authors
- Mathlib users
- Software and hardware verification users

Who is the standard library for?

All Lean users!

- Programmers
  - Programmers writing verified software
  - Metaprogrammers
  - Lean developers
  - Library authors
- Mathlib users
- Software and hardware verification users
- Individuals evaluating Lean for use

Goals of the standard library

Goals of the standard library

Goals of the standard library

Useful for real applications

Goals of the standard library

Useful for real applications

High-quality and polished: comprehensive, consistent, systematic, optimized, verified, testable, tested, documented, interconnected, stable, comprehensible, visible, benchmarked, …

Goals of the standard library

Useful for real applications

High-quality and polished: comprehensive, consistent, systematic, optimized, verified, testable, tested, documented, interconnected, stable, comprehensible, visible, benchmarked, …

Excellent **out-of-the-box experience** for software development and software verification

Setting a high bar

Setting a high bar

Setting a high bar

End goal: make formal verification **economical** and commonplace

Setting a high bar

End goal: make formal verification **economical** and commonplace

Challenge: things don’t just need to be possible, but **easy and productive**

Setting a high bar

End goal: make formal verification **economical** and commonplace

Challenge: things don’t just need to be possible, but **easy and productive**

Need:

Setting a high bar

End goal: make formal verification **economical** and commonplace

Challenge: things don’t just need to be possible, but **easy and productive**

Need:

- No missing material

Setting a high bar

End goal: make formal verification **economical** and commonplace

Challenge: things don’t just need to be possible, but **easy and productive**

Need:

- No missing material
- No inconsistencies or other papercuts

Setting a high bar

End goal: make formal verification **economical** and commonplace

Challenge: things don’t just need to be possible, but **easy and productive**

Need:

- No missing material
- No inconsistencies or other papercuts
- No discoverability issues

Setting a high bar

End goal: make formal verification **economical** and commonplace

Challenge: things don’t just need to be possible, but **easy and productive**

Need:

- No missing material
- No inconsistencies or other papercuts
- No discoverability issues

Need a principled approach to quality!

Tooling

Tooling

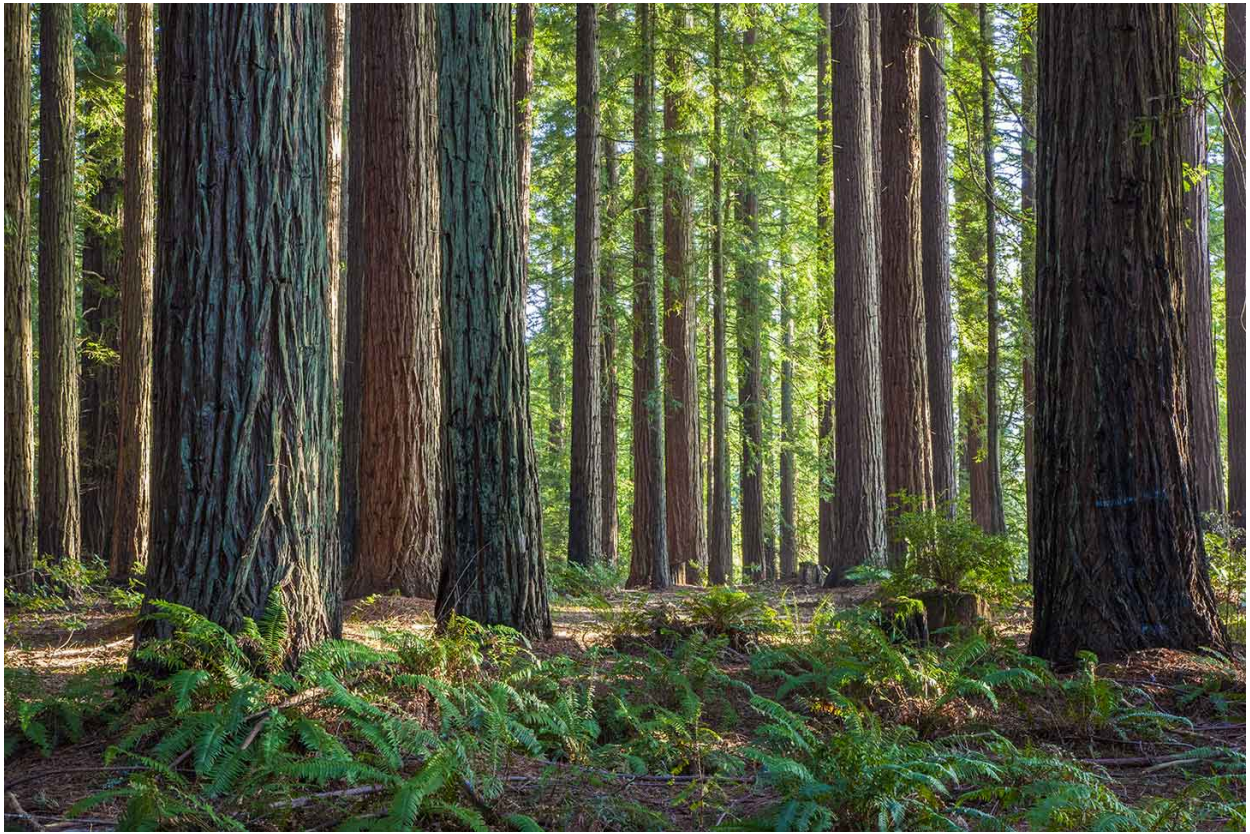
Tooling

Tooling and processes lead to fewer mistakes: CI, linters, code review.

Tooling

Tooling and processes lead to fewer mistakes: CI, linters, code review.

There are gaps!



Beliefs and assumptions

Beliefs and assumptions

Beliefs and assumptions

- Quality assurance will always have a **manual** component

Beliefs and assumptions

- Quality assurance will always have a **manual** component
- Not all rules can be fully formalized, and many rules will have **exceptions**

Beliefs and assumptions

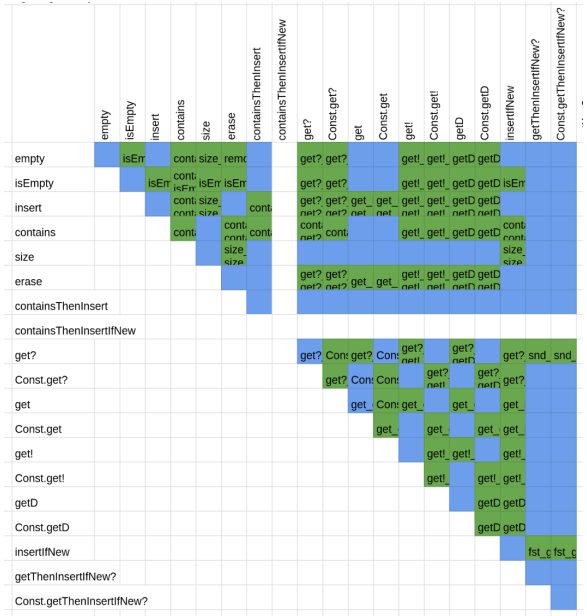
- Quality assurance will always have a **manual** component
- Not all rules can be fully formalized, and many rules will have **exceptions**
- Global **consistency** of the library is desirable

Beliefs and assumptions

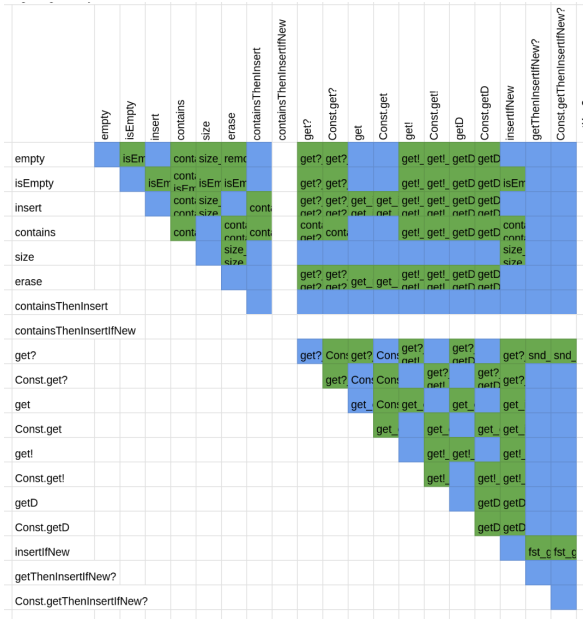
- Quality assurance will always have a **manual** component
- Not all rules can be fully formalized, and many rules will have **exceptions**
- Global **consistency** of the library is desirable
- To understand where we are, we need to be able to **visualize and track** the state of the library

Early experiments

Early experiments



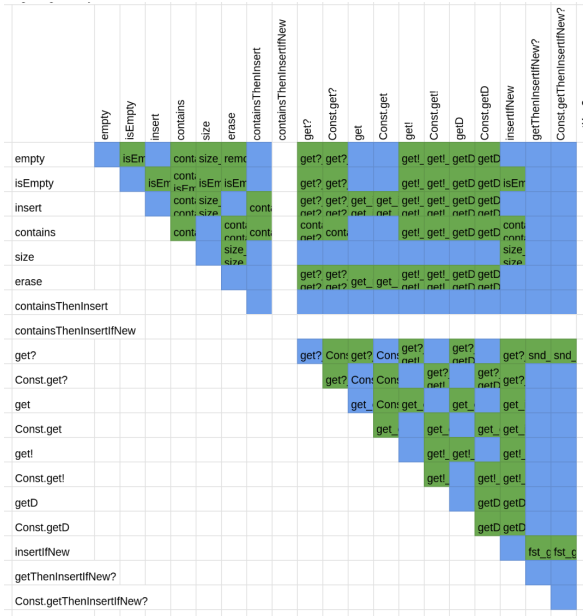
Early experiments



A1 =CONCATENATE("""", JOIN("""","""", A2:A169), """")

	A	B	C	D	E	F
1	"PlainTime","PlainDate","PlainDateTime",[	Std.Time.Plai	Std.Time.Pla	Std.Time.PlainDa	Std.Time.ZonedDa	Std.Time.DateTime
2	PlainTime	X				
3	PlainDate		X			
4	PlainDateTime			X		
5	ZonedDateTime				X	
6	DateTime					X
7	addNanoseconds	X		X	X	X
8	addMilliseconds	X		X	X	X
9	addSeconds	X		X	X	X
10	addMinutes	X		X	X	X
11	addHours	X		X	X	X
12	addDays		X	X	X	X
13	addWeeks		X	X	X	X
14	addMonthsClip		X	X	X	X
15	addMonthsRollOver		X	X	X	X
16	addYearsClip		X	X	X	X
17	addYearsRollOver		X	X	X	X
18	alignedWeekOfMonth		X	X		X
19	atDate	X		X		
20	atTime		X	X		
21	nano					
22	nanosecond	X		X	X	?
23	millisecond	X		X	X	?
24	minute	X		X	X	X
25	second	X		X	X	X
26	hour	X		X	X	X
27	day		X	X	X	X
28	weekOfMonth		X	X	X	X
29	weekOfYear		X	X	X	X
30	weekday		X	X	X	X
31	month		X	X	X	X
32	quarter		X	X	X	X
33	year		X	X	X	X
34	era		X	X	X	X
35	convertZoneRules				X	
36	date			X	X	X
37	time			X	X	?
38	format	X	X	X	X	X
39	fromLeanTime24Hour	X				
40	fromTime12Hour	X				
41	fromTime24Hour	X				
42	fromAmericanDateString		X			
43	fromLeanDateString		X			
44	fromSQLDateString		X			
45	fromAscTimeString			X		X
46	fromDateTimeString			X		
47	fromLeanDateTimeString			X		
48	fromLongDateFormatString			X		X
49	fromDateTimeWithZoneString				X	
50	fromISO8601String				X	
51	fromLeanDateTimeWithZoneString				X	
52	fromRFC822String				X	
53	fromRFC850String				X	
54	toLeanTime24Hour	X				
55	toTime12Hour	X				
56	toTime24Hour	X				

Early experiments



A1 =CONCATENATE("""", JOIN("""","""", A2:A169), """")

	A	B	C	D	E	F
1	"PlainTime","PlainDate","PlainDateTime",[	Std.Time.Plain	Std.Time.Pla	Std.Time.PlainDa	Std.Time.ZonedDa	Std.Time.DateTime
2	PlainTime	X				
3	PlainDate		X			
4	PlainDateTime			X		
5	ZonedDateTime				X	
6	DateTime					X
7	addNanoseconds	X		X	X	X
8	addMilliseconds	X		X	X	X
9	addSeconds	X		X	X	X
10	addMinutes	X		X	X	X
11	addHours	X		X	X	X
12	addDays		X	X	X	X
13	addWeeks		X	X	X	X
14	addMonthsClip		X	X	X	X
15	addMonthsRollOver		X	X	X	X
16	addYearsClip		X	X	X	X
17	addYearsRollOver		X	X	X	X
18	alignedWeekOfMonth		X	X		X
19	atDate	X		X		
20	atTime		X	X		
21	nano					
22	nanosecond	X		X	X	?
23	millisecond	X		X	X	?
24	minute	X		X	X	X
25	second	X		X	X	X
26	hour	X		X	X	X
27	day		X	X	X	X
28	weekOfMonth		X	X	X	X
29	weekOfYear		X	X	X	X
30	weekday		X	X	X	X
31	month		X	X	X	X
32	quarter		X	X	X	X
33	year		X	X	X	X
34	era		X	X	X	X
35	convertZoneRules				X	
36	date			X	X	X
37	time			X	X	?
38	format	X	X	X	X	X
39	fromLeanTime24Hour	X				
40	fromTime12Hour	X				
41	fromTime24Hour	X				
42	fromAmericanDateString		X			
43	fromLeanDateString		X			
44	fromSQLDateString		X			
45	fromAscTimeString			X		X
46	fromDateTimeString			X		
47	fromLeanDateTimeString			X		
48	fromLongDateFormatString			X		X
49	fromDateTimeWithZoneString				X	
50	fromISO8601String				X	
51	fromLeanDateTimeWithZoneString				X	
52	fromRFC822String				X	
53	fromRFC850String				X	
54	toLeanTime24Hour	X				
55	toTime12Hour	X				
56	toTime24Hour	X				

	=CONCATENAT	Fin	BitVec	UInt8	UInt16	UInt32	UInt64	USize	Int8	Int16	Int32	Int64	ISize
120	reduceZeroExtend		X										
121	**Signed arithmetic**												
122	abs		X						?	?	?	?	?
123	neg		X						X	X	X	X	X
124	sdiv		X										
125	smod		X										
126	smtSDiv		X										
127	smtUDiv		X										
128	srem		X										
129	**Unsigned arithmetic**												
130	add	X	X	X	X	X	X	X	X	X	X	X	X
131	div	X		X	X	X	X	X	X	X	X	X	X
132	mod	X		X	X	X	X	X	X	X	X	X	X
133	udiv		X										
134	umod		X										
135	log2	X		X	X	X	X	X					
136	modn	X		X	X	X	X	X					
137	mul	X	X	X	X	X	X	X	X	X	X	X	X
138	sub	X	X	X	X	X	X	X	X	X	X	X	X
139	subNat	X											
140	**Bitwise operations**												
141	land	X		X	X	X	X	X	X	X	X	X	X
142	and		X										
143	lor	X		X	X	X	X	X	X	X	X	X	X
144	or		X										
145	complement			X	X	X	X	X	X	X	X	X	X
146	not		X										
147	xor	X	X	X	X	X	X	X	X	X	X	X	X
148	extractLsb		X										
149	extractLsb'		X										
150	reverse		X										
151	rotateLeft		X										
152	rotateLeftAux		X										
153	rotateRight		X										
154	rotateRightAux		X										
155	shiftConcat		X										
156	shiftLeft	X	X	X	X	X	X	X	X	X	X	X	X
157	shiftRight	X		X	X	X	X	X	X	X	X	X	X
158	sshiftRight		X										
159	sshiftRight'		X										
160	ushiftRight		X										

Next steps

API Manager

Std.DHashMap	Std.HashMap	Std.HashSet
empty	empty	empty
erase	erase	erase
isEmpty	isEmpty	isEmpty
getKey?	getKey?	get?
Membership.mem	Membership.mem	Membership.mem
contains	contains	contains
insertIfNew	insertIfNew	insert
insert	insert	
getD	getD	
get?	get?	
get?	GetElem?.getElem?	

Add Row | Delete Selected Rows | Copy JSON

Step 1: Associate operations across namespaces

Step 2: See which lemmas exist

Step 3: Check lemmas for consistency

Std.HashMap

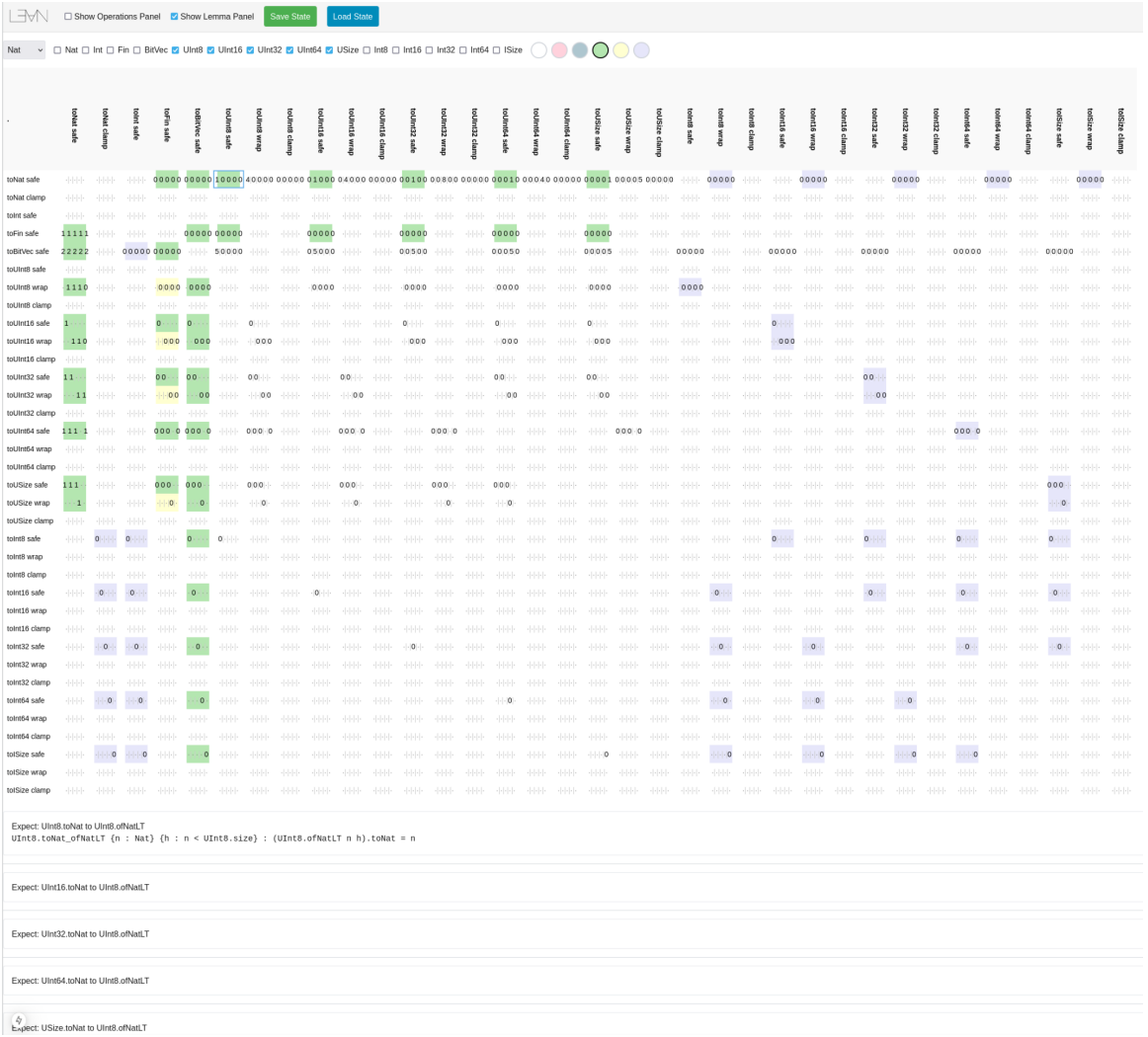
·	empty	erase	isEmpty	getKey?	contains	insertIfNew	insert	getD	get?	GetElem?.getElem
empty	0|0|0	1|1|1	1|1|1	1|1|1	1|1|1	0|0|0	0|0|0	1|1|0	1|0|0	1|1|0
erase		0|0|0	1|1|1	2|2|2	3|3|3	0|0|0	0|0|0	2|2|0	2|0|0	2|2|0
isEmpty			0|0|0	1|1|1	4|4|4	1|1|1	1|1|0	1|1|0	1|0|0	1|1|0
getKey?				0|0|0	4|4|4	1|1|1	2|2|0	0|0|0	0|0|0	0|0|0
contains					1|1|1	4|4|4	3|3|0	2|2|0	4|0|0	4|4|0
insertIfNew						0|0|0	0|0|0	1|1|0	1|0|0	1|1|0
insert							0|0|0	2|2|0	2|0|0	2|2|0
getD								0|1|0	3|0|0	3|3|0
get?									0|0|0	0|1|0
GetElem?.getElem?										0|0|0

```
Std.DHashMap.getKey?_erase.{u, v} {α : Type u} {β : α → Type v} {xt : BEq α} {xt¹ : Hashable α} {m : Std.DHashMap α β}
  [EquivBEq α] [LawfulHashable α] {k a : α} : (m.erase k).getKey? a = if (k == a) = true then none else m.getKey? a
Std.DHashMap.getKey?_erase_self.{u, v} {α : Type u} {β : α → Type v} {xt : BEq α} {xt¹ : Hashable α}
  {m : Std.DHashMap α β} [EquivBEq α] [LawfulHashable α] {k : α} : (m.erase k).getKey? k = none

Std.HashMap.getKey?_erase_self.{u, v} {α : Type u} {β : Type v} {xt : BEq α} {xt¹ : Hashable α} {m : Std.HashMap α β}
  [EquivBEq α] [LawfulHashable α] {k : α} : (m.erase k).getKey? k = none
Std.HashMap.getKey?_erase.{u, v} {α : Type u} {β : Type v} {xt : BEq α} {xt¹ : Hashable α} {m : Std.HashMap α β}
  [EquivBEq α] [LawfulHashable α] {k a : α} : (m.erase k).getKey? a = if (k == a) = true then none else m.getKey? a

Std.HashSet.get?_erase.{u} {α : Type u} {xt : BEq α} {xt¹ : Hashable α} {m : Std.HashSet α} [EquivBEq α]
  [LawfulHashable α] {k a : α} : (m.erase k).get? a = if (k == a) = true then none else m.get? a
Std.HashSet.get?_erase_self.{u} {α : Type u} {xt : BEq α} {xt¹ : Hashable α} {m : Std.HashSet α} [EquivBEq α]
  [LawfulHashable α] {k : α} : (m.erase k).get? k = none
```

Next steps


Show Operations Panel
Show Lemma Panel
Save State
Load State
Nat
Int
Fin
BitVec
UInt8
UInt16
UInt32
UInt64
USize
Int8
Int16
Int32
Int64
ISize
Expect: UInt8.toNat to UInt8.ofNatLT
UInt8.toNat_ofNatLT {n : Nat} {h : n < UInt8.size} : (UInt8.ofNatLT n h).toNat = n
Expect: UInt16.toNat to UInt8.ofNatLT
Expect: UInt32.toNat to UInt8.ofNatLT
Expect: UInt64.toNat to UInt8.ofNatLT
Expect: USize.toNat to UInt8.ofNatLT

The result: Grove

The result: Grove

The result: Grove

Tool for tracking changes to the entire library

The result: Grove

Tool for tracking changes to the entire library

Basic question: **what do we know?** (and when do we no longer know it?)

The result: Grove

Tool for tracking changes to the entire library

Basic question: **what do we know?** (and when do we no longer know it?)

Declaratively and imperatively describe the library and how it should look

The result: Grove

Tool for tracking changes to the entire library

Basic question: **what do we know?** (and when do we no longer know it?)

Declaratively and imperatively describe the library and how it should look

Full power of Lean metaprogramming for extracting state

The result: Grove

Tool for tracking changes to the entire library

Basic question: **what do we know?** (and when do we no longer know it?)

Declaratively and imperatively describe the library and how it should look

Full power of Lean metaprogramming for extracting state

Web UI for analyzing the library

The result: Grove

Tool for tracking changes to the entire library

Basic question: **what do we know?** (and when do we no longer know it?)

Declaratively and imperatively describe the library and how it should look

Full power of Lean metaprogramming for extracting state

Web UI for analyzing the library

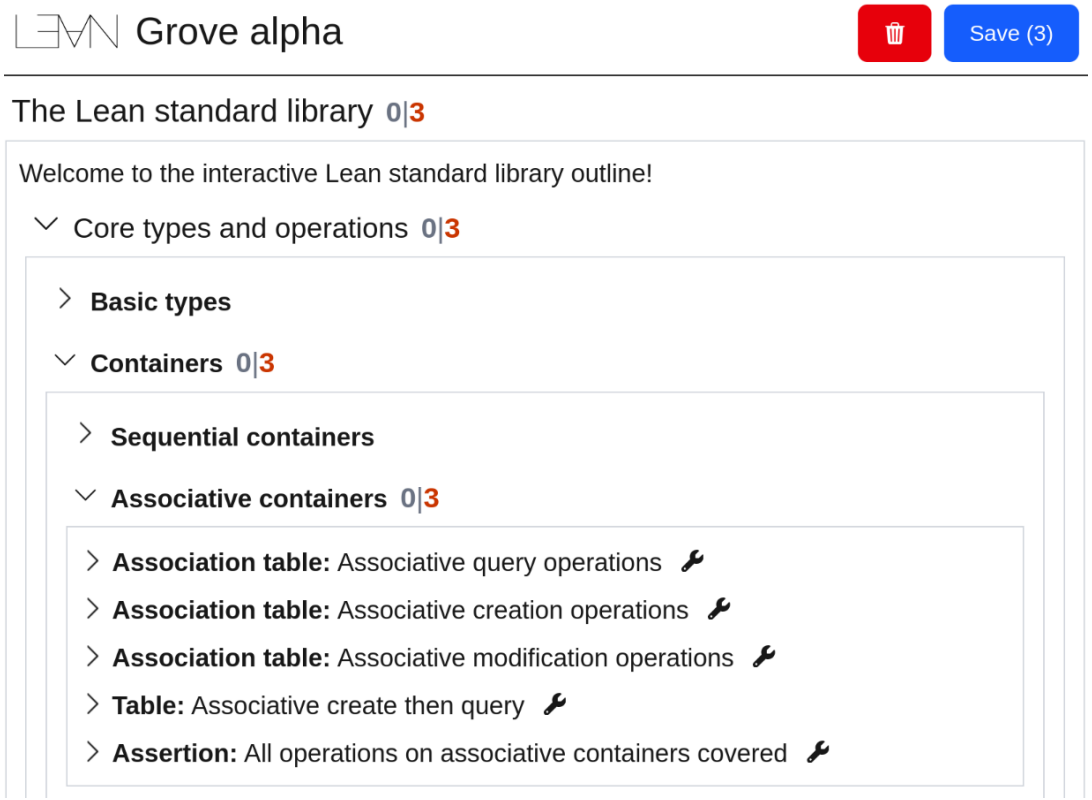
Detects changes in PRs

Grove impressions

```
def associativeContainers : List Lean.Name :=
  [`Std.DHashMap, `Std.DHashMap.Raw, `Std.ExtDHashMap, `Std.DTreeMap,
   `Std.DTreeMap.Raw, `Std.ExtDTreeMap, `Std.HashMap, `Std.HashMap.Raw,
   `Std.ExtHashMap, `Std.TreeMap, `Std.TreeMap.Raw, `Std.ExtTreeMap,
   `Std.HashSet, `Std.HashSet.Raw, `Std.ExtHashSet, `Std.TreeSet,
   `Std.TreeSet.Raw, `Std.ExtTreeSet]

def associativeQueryOperations : AssociationTable .subexpression associativeContainers where
  id := "associative-query-operations"
  title := "Associative query operations"
  description := "Operations that take as input an associative container and return a 'single' pi
  dataSources n :=
    (DataSource.definitionsInNamespace n)
      |>.map Subexpression.declaration
      |>.or (DataSource.getElem n)
```

Grove impressions



Grove impressions

Grove alpha The Lean standard library / Core types and operations / Containers / Associative containers / **Associative query operations**

Title	Std.DHashMap	Std.DHashMap.Raw	Std.ExtDHashMap	Std.DTreeMap	Std.DTreeMap.Raw	Std.ExtDTreeMap	Std.HashMap	Std.HashMap.Raw
isEmpty	Std.DHashMap.isEmpty	Std.DHashMap.Raw.isEmpty	Std.ExtDHashMap.isEmpty	Std.DTreeMap.isEmpty	Std.DTreeMap.Raw.isEmpty	Std.ExtDTreeMap.isEmpty	Std.HashMap.isEmpty	Std.HashMap.Raw.isEmpty
size	Std.DHashMap.size	Std.DHashMap.Raw.size	Std.ExtDHashMap.size	Std.DTreeMap.size	Std.DTreeMap.Raw.size	Std.ExtDTreeMap.size	Std.HashMap.size	Std.HashMap.Raw.size
any				Std.DTreeMap.any	Std.DTreeMap.Raw.any	Std.ExtDTreeMap.any		
all				Std.DTreeMap.all	Std.DTreeMap.Raw.all	Std.ExtDTreeMap.all		
getD	Std.DHashMap.getD	Std.DHashMap.Raw.getD	Std.ExtDHashMap.getD	Std.DTreeMap.getD	Std.DTreeMap.Raw.getD	Std.ExtDTreeMap.getD	Std.HashMap.getD	Std.HashMap.Raw.getD
getElem	Std.DHashMap.get	Std.DHashMap.Raw.Const.get	Std.ExtDHashMap.get	Std.DTreeMap.get	Std.DTreeMap.Raw.get	Std.ExtDTreeMap.get	Std.HashMap[·]	Std.HashMap.Raw[·]
isSingleton						Std.ExtDTreeMap.isSingleton		

	isEmpty	size	getD	getElem
empty	330220	110000	110000	000000
ofList	110000	222112	222222	101111
emptyCollection	653113	225115	221201	002000

Grove impressions

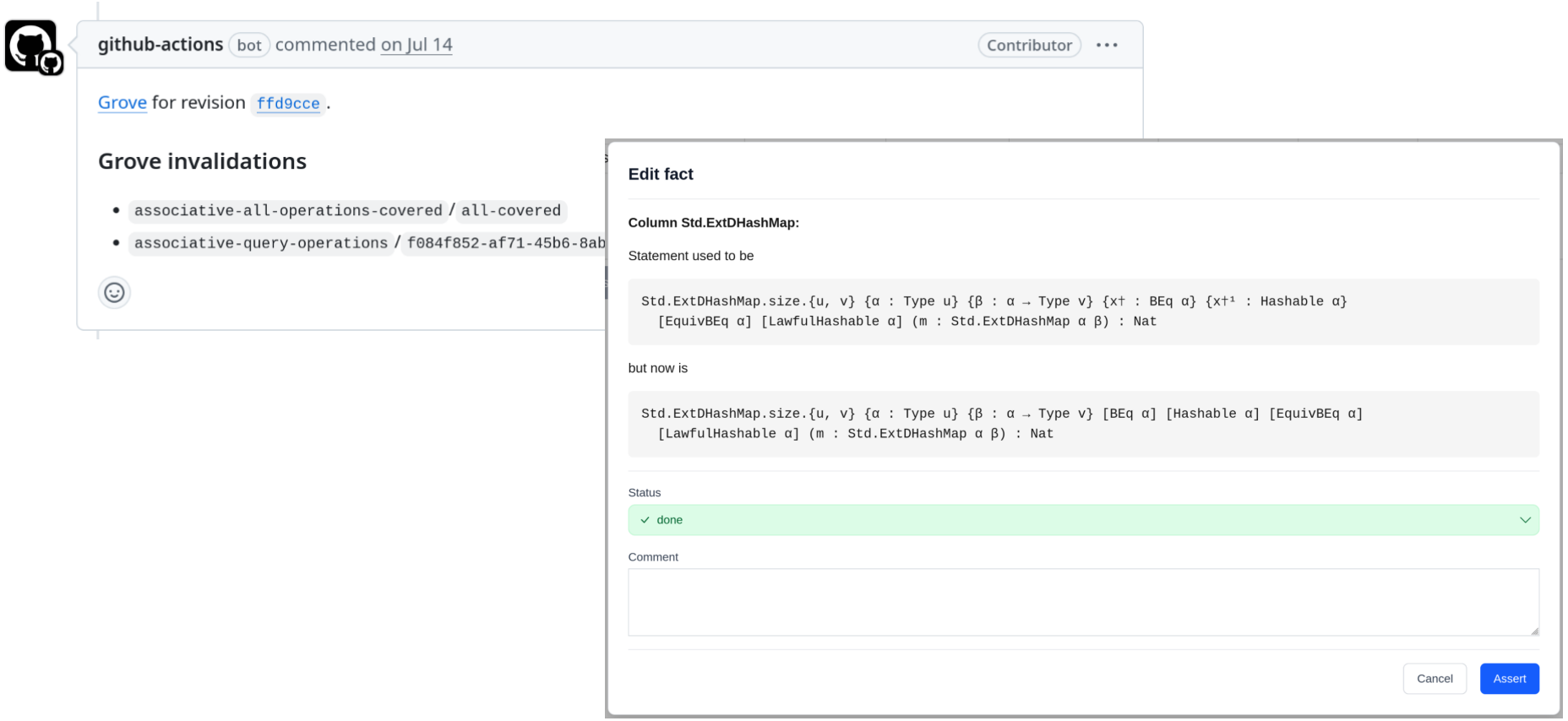
github-actions bot commented on Jul 14 Contributor ···

Grove for revision `ffd9cce`.

Grove invalidations

- `associative-all-operations-covered` / `all-covered`
- `associative-query-operations` / `f084f852-af71-45b6-8ab3-d251a8144f72`

Grove impressions



Grove impressions

```
def «5ceaa26a-d2cb-4df3-9ac8-b5c11db2ae9d:::01f88623-fa5f-4380-9772-b30f2fec5c94:::Std.DHashMap::Std.DHashMap.Raw::Std.ExtDHashMap::Std.DTreeMap::Std.DTreeMap.Raw::Std.ExtD
  widgetId := "associative-create-then-query"
  factId := "5ceaa26a-d2cb-4df3-9ac8-b5c11db2ae9d:::01f88623-fa5f-4380-9772-b30f2fec5c94:::Std.DHashMap::Std.DHashMap.Raw::Std.ExtDHashMap::Std.DTreeMap::Std.DTreeMap.Raw::
  rowAssociationId := "5ceaa26a-d2cb-4df3-9ac8-b5c11db2ae9d"
  columnAssociationId := "01f88623-fa5f-4380-9772-b30f2fec5c94"
  selectedLayers := #["Std.DHashMap", "Std.DHashMap.Raw", "Std.ExtDHashMap", "Std.DTreeMap", "Std.DTreeMap.Raw", "Std.ExtDTreeMap", ]
  layerStates := #[
    {
      layerIdentifier := "Std.DHashMap"
      rowState :=

        some ("app (EmptyCollection.emptyCollection) (Std.DHashMap*)", Grove.Framework.Subexpression.State.predicate
          { key := "app (EmptyCollection.emptyCollection) (Std.DHashMap*)", displayShort := "∅" })

      columnState :=

        some ("Std.DHashMap.isEmpty", Grove.Framework.Subexpression.State.declaration
          (Grove.Framework.Declaration.def
            { name := `Std.DHashMap.isEmpty,
              renderedStatement := "Std.DHashMap.isEmpty.{u, v} {α : Type u} {β : α → Type v} {x† : BEq α} {x†' : Hashable α}\n  (m : Std.DHashMap α β) : Bool",
              isDeprecated := false }))

        selectedCellStates := #[
          ("Std.DHashMap.isEmpty_empty", Grove.Framework.Declaration.thm
{ name := `Std.DHashMap.isEmpty_empty,
  renderedStatement := "Std.DHashMap.isEmpty_empty.{u, v} {α : Type u} {β : α → Type v} {x† : BEq α} {x†' : Hashable α} :\n  ∅.isEmpty = true",
  isSimp := true,
  isDeprecated := false })
,
        ]
    },
    {
      layerIdentifier := "Std.DHashMap.Raw"
      rowState :=

        some ("app (EmptyCollection.emptyCollection) (Std.DHashMap.Raw*)", Grove.Framework.Subexpression.State.predicate
          { key := "app (EmptyCollection.emptyCollection) (Std.DHashMap.Raw*)", displayShort := "∅" })

      columnState :=

        some ("Std.DHashMap.Raw.isEmpty", Grove.Framework.Subexpression.State.declaration
          (Grove.Framework.Declaration.def
            { name := `Std.DHashMap.Raw.isEmpty,
              renderedStatement := "Std.DHashMap.Raw.isEmpty.{u, v} {α : Type u} {β : α → Type v} (m : Std.DHashMap.Raw α β) : Bool",
              isDeprecated := false }))

        selectedCellStates := #[
          ("Std.DHashMap.Raw.isEmpty_emptyc", Grove.Framework.Declaration.thm
{ name := `Std.DHashMap.Raw.isEmpty_emptyc,
  renderedStatement := "Std.DHashMap.Raw.isEmpty_emptyc.{u_1, u_2} {α : Type u_1} {β : α → Type u_2} [BEq α] [Hashable α] :\n  ∅.isEmpty = true",
  isSimp := false,
  isDeprecated := true })
,
        ]
    },
    {
      layerIdentifier := "Std.ExtDHashMap"
      rowState :=

        some ("app (EmptyCollection.emptyCollection) (Std.ExtDHashMap*)", Grove.Framework.Subexpression.State.predicate
```

Grove architecture