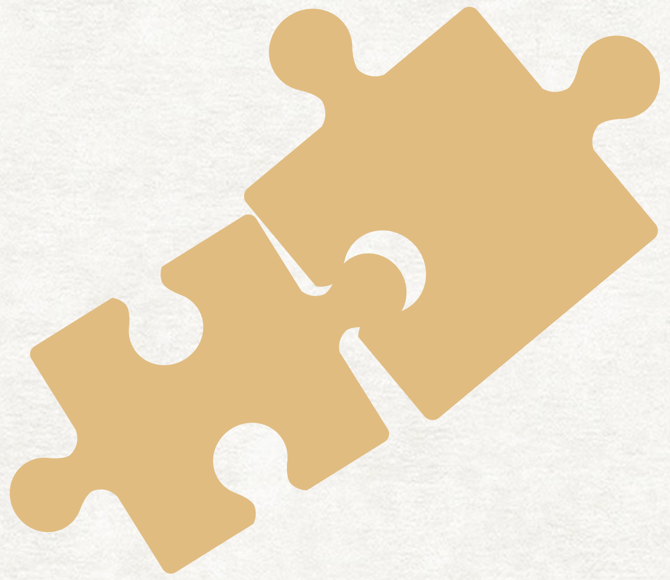
FORMAL REASONING USING DISTRIBUTED ASSERTIONS



FARAH AL WARDANI
KAUSTUV CHAUDHURI
DALE MILLER

LIX, INRIA SACLAY

Workshop on Libraries of Formal Proofs and Natural Mathematical Language

EuroProofNet Joint WG4-WG5 meeting, Cambridge, England

September 8, 2023

Many different systems

Common (general) goal

Separately growing environments

THE PREVALENT VIEW

SYSTEMS AS SEPARATE ENVIRONMENTS

Many different systems

Common (general) goal

Separately growing environments

THE PREVALENT VIEW

SYSTEMS AS SEPARATE ENVIRONMENTS

Side (major) effects?

Redundant information and effort

Disconnected information and processes

Inefficiency

Lost benefits of connection (modularity..)

Ivan Ayvazovsky, Public domain, via Wikimedia Commons

THE QUEST FOR INTEROPERABILITY

THE QUEST FOR INTEROPERABILITY

one-to-one integration

Re-checkable proof certificates

Or

External system as trusted procedure

Examples:

Flyspeck project, SAT and SMT in Coq, etc

THE QUEST FOR INTEROPERABILITY

one-to-one integration

Re-checkable proof certificates

Or

External system as trusted procedure

Examples:

Flyspeck project, SAT and SMT in Coq, etc

Observations

- Limited to specific systems
- Not always feasible

THE QUEST FOR INTEROPERABILITY

many-to-one integration

(aim: universal interoperability)

Evidential tool bus

Dedukti, MMT

- Logics and systems as theories in one trusted system
- Translating and combining libraries (theorems and proofs) into one trusted system

TLAPS

- several systems trusted by main system

THE QUEST FOR INTEROPERABILITY

many-to-one integration

(aim: universal interoperability)

Evidential tool bus

Dedukti, MMT

- Logics and systems as theories in one trusted system
- Translating and combining libraries (theorems and proofs) into one trusted system

TLAPS

- several systems trusted by main system

Observations

- Assume a central framework/language/system

THE QUEST FOR INTEROPERABILITY

standard formats and languages

Bridges between systems

OMDoc/MMT, TPTP

THE QUEST FOR INTEROPERABILITY

standard formats and languages

Bridges between systems

OMDoc/MMT, TPTP

Observations

- useful and needed BUT not enough
- no easy consensus

THE QUEST FOR INTEROPERABILITY

universal libraries (incorporating different systems)

QED Manifesto, Formal Abstracts,

Logosphere, Logipedia …

Observations

- Many proposed libraries (natural)
- But disconnected

THE QUEST FOR INTEROPERABILITY

many approaches,
many dimensions

Additional layer of disconnected environments

Existing systems and libraries integration

Transporting and rechecking proofs problematic

THE QUEST FOR INTEROPERABILITY

many approaches, many dimensions

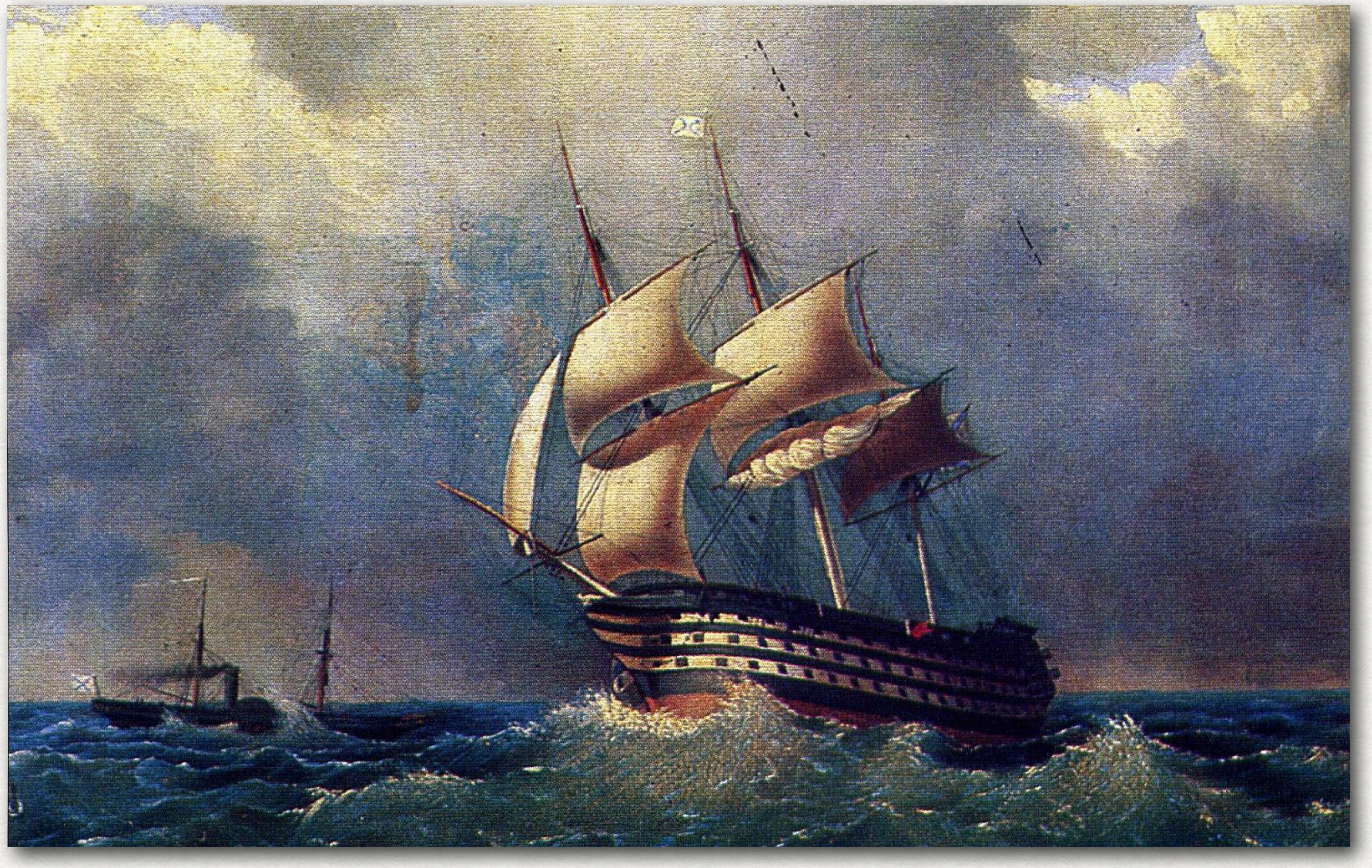
Additional layer of disconnected environments

Existing systems and libraries integration

Transporting and rechecking proofs problematic

New system:

- Where to go?
- Efforts? dependent on other system and community?
- Can and will (or even want to) be expressible in a pre-defined central logical framework/meta-language?


Ivan Ayvazovsky, Public domain, via Wikimedia Commons

EXPLORING A DIFFERENT DIMENSION

EXPLORING A DIFFERENT DIMENSION

Why not start from simple needs?

~~Integrate systems, libraries~~

EXPLORING A DIFFERENT DIMENSION

starting goal

EXPLORING A DIFFERENT DIMENSION

starting goal

~~Integrate systems, libraries~~

Use a theorem proved in one development

as a lemma

in another development

Theorem. For $n \in \mathbb{N}$, $\operatorname{fib}(n) = n^2$ if and only if $n \in \{0, 1, 12\}$, where $\operatorname{fib}(n)$ stands for the nth Fibonacci number defined as: $\operatorname{fib}(0) \triangleq 0$, $\operatorname{fib}(1) \triangleq 1$, and $\operatorname{fib}(n+2) \triangleq \operatorname{fib}(n+1) + \operatorname{fib}(n)$.

EXPLORING A DIFFERENT DIMENSION

EXAMPLE

Theorem. For $n \in \mathbb{N}$, $\mathrm{fib}(n) = n^2$ if and only if $n \in \{0, 1, 12\}$, where $\mathrm{fib}(n)$ stands for the nth Fibonacci number defined as: $\mathrm{fib}(0) \triangleq 0$, $\mathrm{fib}(1) \triangleq 1$, and $\mathrm{fib}(n+2) \triangleq \mathrm{fib}(n+1) + \mathrm{fib}(n)$.

In Abella?

EXPLORING A DIFFERENT DIMENSION

EXAMPLE

Theorem. For $n \in \mathbb{N}$, $\mathrm{fib}(n) = n^2$ if and only if $n \in \{0, 1, 12\}$, where $\mathrm{fib}(n)$ stands for the nth Fibonacci number defined as: $\mathrm{fib}(0) \triangleq 0$, $\mathrm{fib}(1) \triangleq 1$, and $\mathrm{fib}(n+2) \triangleq \mathrm{fib}(n+1) + \mathrm{fib}(n)$.

In Abella?

EXPLORING A DIFFERENT DIMENSION

EXAMPLE

n in {0, 1, 12} --> fib(n) = n^2 -- easy in Abella

Theorem. For $n \in \mathbb{N}$, $\mathrm{fib}(n) = n^2$ if and only if $n \in \{0, 1, 12\}$, where $\mathrm{fib}(n)$ stands for the nth Fibonacci number defined as: $\mathrm{fib}(0) \triangleq 0$, $\mathrm{fib}(1) \triangleq 1$, and $\mathrm{fib}(n+2) \triangleq \mathrm{fib}(n+1) + \mathrm{fib}(n)$.

In Abella?

n in {0, 1, 12} --> fib(n) = n^2 -- easy in Abella

fib(n) = n^2 --> n in {0, 1, 12} ????

EXPLORING A DIFFERENT DIMENSION

EXAMPLE

Theorem. For $n \in \mathbb{N}$, $\mathrm{fib}(n) = n^2$ if and only if $n \in \{0, 1, 12\}$, where $\mathrm{fib}(n)$ stands for the nth Fibonacci number defined as: $\mathrm{fib}(0) \triangleq 0$, $\mathrm{fib}(1) \triangleq 1$, and $\mathrm{fib}(n+2) \triangleq \mathrm{fib}(n+1) + \mathrm{fib}(n)$.

In Abella?

n in {0, 1, 12} --> fib(n) = n^2 -- easy in Abella

fib(n) = n^2 --> n in {0, 1, 12} ????

EXPLORING A DIFFERENT DIMENSION

EXAMPLE

need this LEMMA

Lemma. For $n \in \mathbb{N}$, if $n \geq 13$, then $\mathrm{fib}(n) > n^2$.

Not easily proved in Abella 2.0

Theorem. For $n \in \mathbb{N}$, $\mathrm{fib}(n) = n^2$ if and only if $n \in \{0, 1, 12\}$, where $\mathrm{fib}(n)$ stands for the nth Fibonacci number defined as: $\mathrm{fib}(0) \triangleq 0$, $\mathrm{fib}(1) \triangleq 1$, and $\mathrm{fib}(n+2) \triangleq \mathrm{fib}(n+1) + \mathrm{fib}(n)$.

EXPLORING A DIFFERENT DIMENSION

EXAMPLE

In Abella?

n in {0, 1, 12} --> fib(n) = n^2 -- easy in Abella

fib(n) = n^2 --> n in {0, 1, 12} ????

need this LEMMA

Lemma. For $n \in \mathbb{N}$, if $n \geq 13$, then $\mathrm{fib}(n) > n^2$.

Not easily proved in Abella 2.0

-- why not reuse from Coq?

EXPLORING A DIFFERENT DIMENSION

EXAMPLE

LEMMA, in Coq

Lemma. For $n \in \mathbb{N}$, if $n \geq 13$, then $\mathrm{fib}(n) > n^2$.

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

Theorem. For $n \in \mathbb{N}$, $\mathrm{fib}(n) = n^2$ if and only if $n \in \{0, 1, 12\}$, where $\mathrm{fib}(n)$ stands for the nth Fibonacci number defined as: $\mathrm{fib}(0) \triangleq 0$, $\mathrm{fib}(1) \triangleq 1$, and $\mathrm{fib}(n+2) \triangleq \mathrm{fib}(n+1) + \mathrm{fib}(n)$.

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

HOW?

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

HOW?

Get the theorem from Coq

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

HOW?

Get the theorem from Coq

Translate it into Abella language

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

HOW?

Get the theorem from Coq

Translate it into Abella language

Trust the (whole) development

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

Get the theorem from Coq?

Some form of information representation, storage, and retrieval

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

Translate it into Abella language?

Incorporating translation aspects into information representation solution

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

Trust the (whole) development?

1. Trusting the Coq development
2. Trusting the Abella development

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

1. Trusting the Coq development

Transport Coq proof into Abella?

Maybe not feasible, so don't assume it

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

1. Trusting the Coq development

Trust the proof as checked in Coq?

need some form of stamp: verifiable info

"I, some user, have indeed proved this theorem (`fib_square_above`), in Coq"

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

1. Trusting the Coq development

Trust the proof as checked in Coq?

need some form of stamp:

ASSERTION

"I, some user, have indeed proved this theorem (`fib_square_above`), in Coq"

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

2. Trusting the Abella development

Trust the proof as checked in Abella?

(if abella user, normal trusting of abella kernel)

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

End goal of example: reuse whole development

Trust the (whole) development?

"I, some user, have indeed proved this theorem (**fib_squares**), in Abella, depending on (**fib_square_above**)"

? ASSERTION-1

"I, some user, have indeed proved this theorem (**fib_square_above**), in Coq"

ASSERTION-0

Ivan Ayvazovsky, Public domain, via Wikimedia Commons

INTRODUCING DAMF

The Distributed Assertion Management Framework

INTRODUCING, DAMF

goal

Establish distributed ground for exchange

- before combining pre-existing libraries and systems

New system: small effort to connect

- at least regarding ability to make its assertions available for others many systems, users

no central system, repository

reuse, publish assertions globally

Establish distributed ground for exchange

before combining pre-existing libraries and systems

INTRODUCING, DAMF

goal

New system: small effort to connect

at least regarding ability to make its assertions available for others

The poor poet, Carl Spitzweg. Public domain, via Wikimedia Commons

DESIGN & IMPLEMENTATION

The poor poet, Carl Spitzweg. Public domain, via Wikimedia Commons

DESIGN & IMPLEMENTATION

outset
concerns

OUTSET CONCERNS

assertion
presentation

Consider diverse tools, users

- Each user could use mode/tool differently
- Agents, in a distributed environment

Consider diverse tools, users

Each user could use mode/tool differently

Agents, in a distributed environment

OUTSET CONCERNS

assertion presentation

ASSERTION

K (agent) says C (claim)

Consider diverse tools, users

Each user could use mode/tool differently

Agents, in a distributed environment

OUTSET CONCERNS

assertion presentation

ASSERTION

K (agent) says C (claim)

"proved theorem in mode",
"proved theorem in mode,
depending on another theorem"

Consider diverse tools, users

Each user could use mode/tool differently

Agents, in a distributed environment

OUTSET CONCERNS

assertion
presentation

ASSERTION

K (agent) says C (claim)

"proved theorem in mode",
"proved theorem in mode,
depending on another theorem"

public-key cryptography

A robust way of verification

Information need to be:

clearly specified, globally accessible

OUTSET CONCERNS

information presentation, storage, retrieval

OUTSET CONCERNS

information presentation, storage, retrieval

Information need to be:

clearly specified, globally accessible

Global repository of objects

How to represent such a global repository?

OUTSET CONCERNS

information presentation, storage, retrieval

distributed

devoid of naming conflicts

OUTSET CONCERNS

information presentation, storage, retrieval

Usual web-addressing scheme (URLs)?

- distributed - YES
  - BUT, consider:
  - data inaccessibility or modification
  - distributed, adversarial setting
- devoid of naming conflicts - NO

Content-addressing scheme

OUTSET CONCERNS

information presentation, storage, retrieval

OUTSET CONCERNS

information presentation, storage, retrieval

Content-addressing scheme

InterPlanetary File System (IPFS):

> "a modular suite of protocols for organizing and transferring data, designed from the ground up with the principles of content addressing and peer-to-peer networking"

IPLD (InterPlanetary Linked Data)

to present linked data in IPFS

Data persistence, integrity, deduplication, etc

Example Assertion Object

Name: IPFS Content Identifier (cid) - hash

`bafyreiek2t75whn7gi6ygrymegguescqi4iudjj56uitnij775u2e2j3nu`

Example Assertion Object

Name: IPFS Content Identifier (cid) - hash

bafyreiek2t75whn7gi6ygrymegguescqi4iudjj56uitnij775u2e2j3nu


```
{ "format":"assertion",
  "agent":"-----BEGIN PUBLIC KEY-----\nMFIwEAYHKoZ... \n-----END PUBLIC KEY-----\n",
  "claim": {"/":"bafyreibvtxzqhvht5rfxpw3rkgx3xliotvjsgqps2y..."},
  "signature":"3040021e10db76a6606d7a813747849028c79e52eea3976fe..." }
```

The poor poet, Carl Spitzweg. Public domain, via Wikimedia Commons

DESIGN & IMPLEMENTATION

Information:
a global, resilient,
connected view

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

Presenting Assertions:

Assertion:

claim

agent

signature

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

Presenting Assertions:

Assertion:

claim?

agent

signature

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

Presenting Assertions:

Claim:

"Produced this theorem in Coq"

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

Presenting Assertions:

Claim:

"Produced this theorem in Coq"

Production object

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

Presenting Assertions:

Claim:

"Produced this theorem in Coq"

Production object

Formula, mode?

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

Presenting Assertions:

Claim:

"Produced this theorem in Coq"

Production object

Formula, mode?

enough?

Presenting Assertions:

Claim:

need to track dependency on a locally Unproved lemma

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

"Produced this theorem in Abella depending on ..."

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

Presenting Assertions:

Claim:

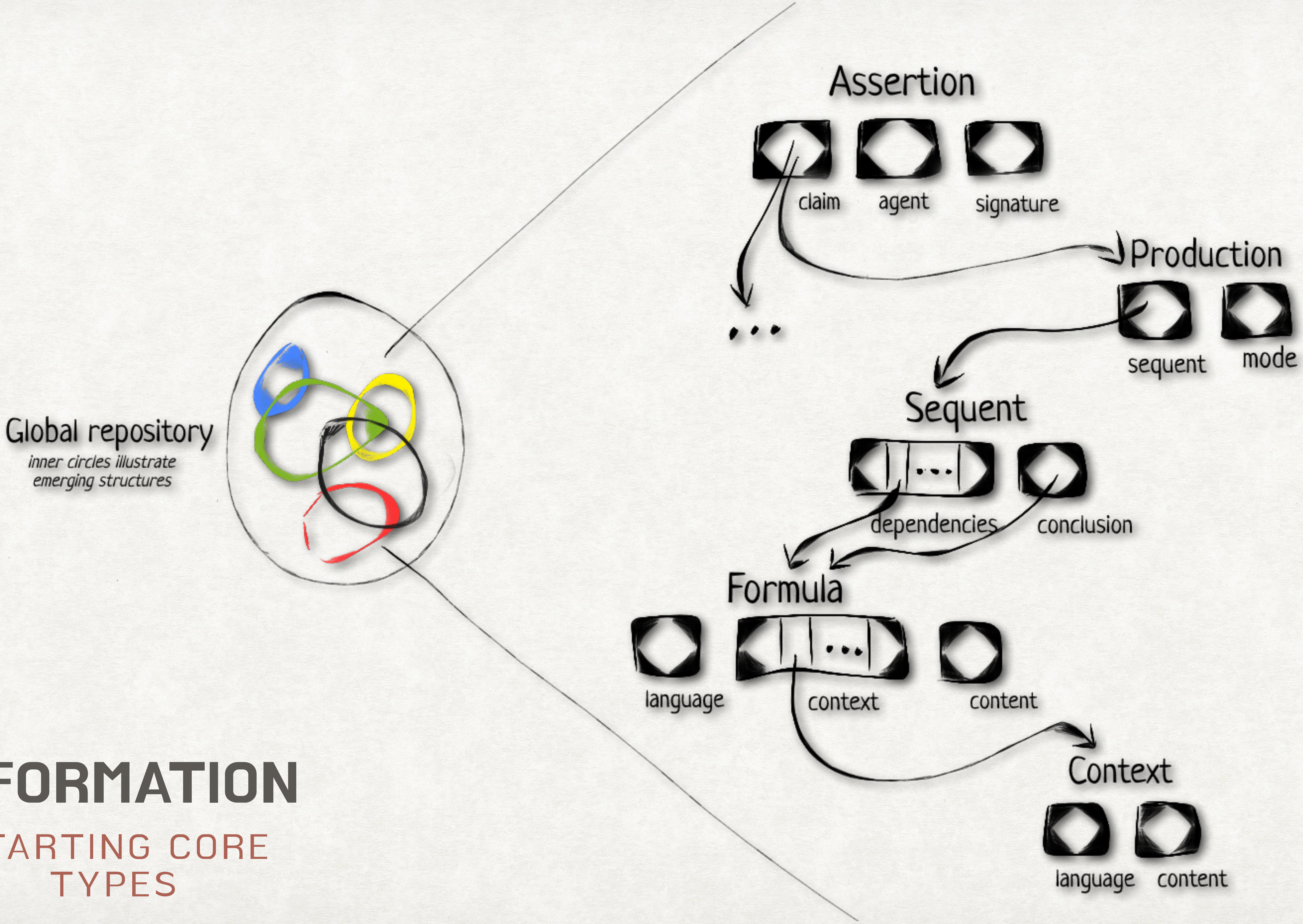
need to track dependency on a locally Unproved lemma

"Produced this theorem in Abella depending on ..."

Production object

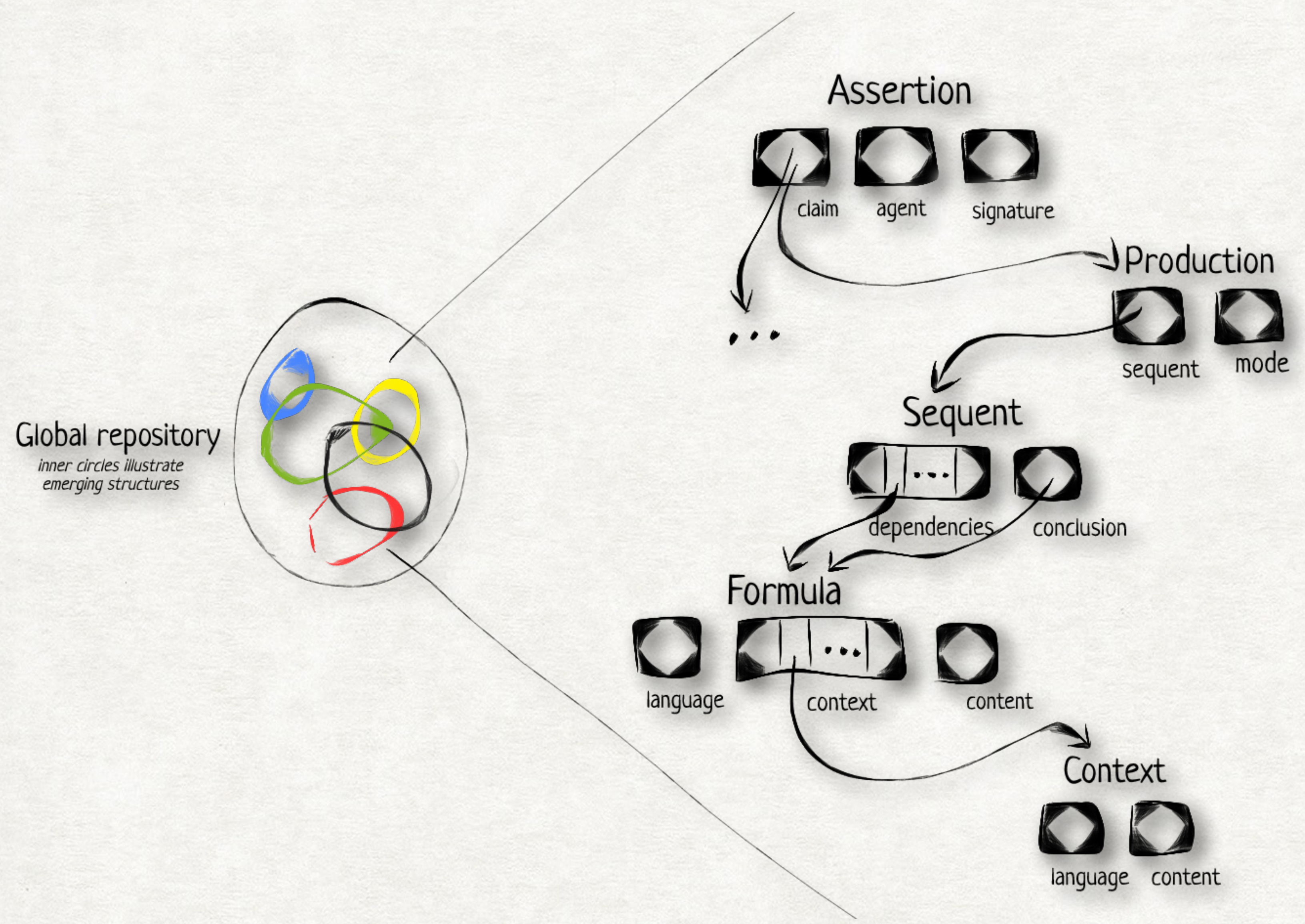
~~Formula, mode~~?

Sequent, mode



INFORMATION

STARTING CORE TYPES



Available in global repository

Known and fixed meanings

Produced by agents

Independent of consumers criteria of what and how

INFORMATION

STARTING CORE TYPES

```
{ "format":"assertion",
  "agent":"-----BEGIN PUBLIC KEY-----\nMFIwEAYHKoZ... \n-----END PUBLIC KEY-----\n",
  "claim": {"/":"bafyreibvtxzqhvht5rfxpw3rkgx3xliotvjsgqps2y..."},
  "signature":"3040021e10db76a6606d7a813747849028c79e52eea3976fe..." }
```

INFORMATION

STARTING CORE TYPES

```
{ "format":"assertion",
  "agent":"-----BEGIN PUBLIC KEY-----\nMFIwEAYHKoZ... \n-----END PUBLIC KEY-----\n",
  "claim": {"/":"bafyreibvtxzqhvht5rfxpw3rkgx3xliotvjsgqps2y..."},
  "signature":"3040021e10db76a6606d7a813747849028c79e52eea3976fe..." }
```

```
{ "format":"annotated-production",
  "production": {"/":"bafyre..."},
  "annotation": {"/":"bafyre..."} }
```

INFORMATION

STARTING CORE TYPES

{ "format":"assertion",
  "agent":"-----BEGIN PUBLIC KEY-----\nMFIwEAYHKoZ... \n-----END PUBLIC KEY-----\n",
  "claim": {"/":"bafyreibvtxzqhvht5rfxpw3rkgx3xliotvjsgqps2y..."},
  "signature":"3040021e10db76a6606d7a813747849028c79e52eea3976fe..." }

{ "format":"annotated-production",
  "production": {"/":"bafyre..."},
  "annotation": {"/":"bafyre..."} }

{ "format":"production",
  "sequent": {"/":"bafyre..."},
  "mode": {"/":"bafyre..."} }

And so on ...

INFORMATION

STARTING CORE TYPES

The poor poet, Carl Spitzweg. Public domain, via Wikimedia Commons

DESIGN & IMPLEMENTATION

Dispatch: an intermediary tool

DISPATCH

an intermediary tool

Package up common procedures between agents

- interaction with IPFS
- type validation of objects

Facilitating participation of systems in DAMF

DISPATCH

an intermediary tool

Package up common procedures between agents

- interaction with IPFS
- type validation of objects

Facilitating participation of systems in DAMF

- Dispatch: "an" intermediary tool between agents and the global repository

DISPATCH

an intermediary tool

Package up common procedures between agents

- interaction with IPFS
- type validation of objects

Facilitating participation of systems in DAMF

- Dispatch: "an" intermediary tool between agents and the global repository

First two functionalities

- publish
- get

```json
{"format": "assertion",
 "agent": "localAgent",
 "claim": {
   "format": "annotated-production",
   "annotation": . . .,
   "production": {
     "mode": "damf:bafyreihnx2. . .",
     "sequent": {
       "conclusion": "plus_comm",
       "dependencies": [ "damf:bafyreihw6g. . .", "plus_succ" ] } } },
 "formulas": {
   "plus_comm": {
     "language": "damf:bafyreidyts. . .",
     "content": ": forall M N K, nat K → . . .",
     "context": ["plus"] },
   "plus_succ": {
     "language": "damf:bafyreidyts. . ...",
     "content": ": forall M N K, . . .",
     "context": ["plus"] } },
 "contexts": {
   "plus": {
     "language": "damf:bafyreidyts. . ...",
     "content": [
       "Kind nat type.", "Type z nat.", "Type s nat → nat.",
       "Define plus : nat → nat → prop by . . .." ] } } }
```


>> dispatch publish >>

```json
{"format": "assertion",
 "agent": "localAgent",
 "claim": {
   "format": "annotated-production",
   "annotation": . . .,
   "production": {
     "mode": "damf:bafyreihnx2. . .",
     "sequent": {
       "conclusion": "plus_comm",
       "dependencies": [ "damf:bafyreihw6g. . .", "plus_succ" ] } } },
 "formulas": {
   "plus_comm": {
     "language": "damf:bafyreidyts. . .",
     "content": ": forall M N K, nat K → . . .",
     "context": ["plus"] },
   "plus_succ": {
     "language": "damf:bafyreidyts. . ...",
     "content": ": forall M N K, . . .",
     "context": ["plus"] } },
 "contexts": {
   "plus": {
     "language": "damf:bafyreidyts. . ...",
     "content": [
       "Kind nat type.", "Type z nat.", "Type s nat → nat.",
       "Define plus : nat → nat → prop by . . .." ] } } }
```



```json
{ "format": "assertion",
   "claim": {
     "/": "bafyreibvtxzqhvht5rfxpw3rkgx3xliotvjs. . ." },
   "agent": "-----BEGIN PUBLIC KEY-----\nMFIwEA. . .",
   "signature": "3040021e10db76a6606d7a813747849028c79e. . ." }
```


CID

>> dispatch publish >>

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

bafyreifswwilvznjy76kfs3f2wotnjpllok64n...

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

bafyreifswwilvznjy76kfs3f2wotnjpllok64n...

```
> ipfs dag get
bafyreifswwilvznjy76kfs3f2wot.../claim/
production/sequent/conclusion/content
```

"forall n, 13 <= n -> n ^ 2 < fib n"

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

[bafyreifswwilvznjy76kfs3f2wotnjpllok64n...]

> ipfs dag get bafyreifswwilvznjy76kfs3f2wot.../claim/production/sequent/conclusion/content

```
"forall n, 13 <= n -> n ^ 2 < fib n"
```

?

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

NOW, after we got the Coq Assertion ready, how to use it in the Abella development?


The poor poet, Carl Spitzweg. Public domain, via Wikimedia Commons

DESIGN & IMPLEMENTATION

DAMF-aware system:
Abella-DAMF

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

bafyreifswwilvznjy76kfs3f2wotnjpllok64n...

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

bafyreifswwilvznjy76kfs3f2wotnjpllok64n...

Importing LEMMA into Abella

```
Import "damf:bafyreifswwil. . ." as
Theorem fib_square_above: forall x, nat x ->
  leq (s^13 z) x ->
  forall y, times x x y ->
  forall u, fib x u -> lt y u.
```

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

bafyreifswwilvznjy76kfs3f2wotnjpllok64n...

Importing LEMMA into Abella

```
Import "damf:bafyreifswwil. . ." as
Theorem fib_square_above: forall x, nat x ->
  leq (s^13 z) x ->
  forall y, times x x y ->
  forall u, fib x u -> lt y u.
```

```
>> dispatch get (imported assertion)
```

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

bafyreifswwilvznjy76kfs3f2wotnjpllok64n...

Importing LEMMA into Abella

```
Import "damf:bafyreifswwil. . ." as
Theorem fib_square_above: forall x, nat x ->
  leq (s^13 z) x ->
  forall y, times x x y ->
  forall u, fib x u -> lt y u.
```

```
>> dispatch get (imported assertion)
>> dispatch publish (adapter assertion)
```

```
dependencies: cid of fib_square_above(Coq)
conclusion: cid of fib_square_above(Abella)
```

LEMMA, in Coq

```
Theorem fib_square_above: forall n, 13 <= n -> n ^ 2 < fib n.
Proof.
  ....
Qed.
```

bafyreifswwilvznjy76kfs3f2wotnjpllok64n...

Importing LEMMA into Abella

```
Import "damf:bafyreifswwil. . ." as
Theorem fib_square_above: forall x, nat x ->
  leq (s^13 z) x ->
  forall y, times x x y ->
  forall u, fib x u -> lt y u.
```

```
>> dispatch get (imported assertion)
>> dispatch publish (adapter assertion)
```

```
dependencies: cid of fib_square_above(Coq)
conclusion: cid of fib_square_above(Abella)
```

THEOREM, in Abella

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

ADAPTERS

Adapters as a general concept: many purposes

translation of formulas, normalizing, renaming, resolving conflicts, logical operations (instantiation, unfolding ..)

Manual or automatic construction

Non-harmful possibility of exploring combinations

THEOREM, in Abella

```
Import "damf:bafyreifswwil. . ." as
Theorem fib_square_above: forall x, nat x ->
  leq (s^13 z) x ->
  forall y, times x x y ->
  forall u, fib x u -> lt y u.
```

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

Now, it's possible to publish the Abella development (assertion)

THEOREM, in Abella

```
Import "damf:bafyreifswwil. . ." as
Theorem fib_square_above: forall x, nat x ->
  leq (s^13 z) x ->
  forall y, times x x y ->
  forall u, fib x u -> lt y u.
```

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

```
abella publishing mode >> dispatch publish
```

THEOREM, in Abella

```
Import "damf:bafyreifswwil. . ." as
Theorem fib_square_above: forall x, nat x ->
  leq (s^13 z) x ->
  forall y, times x x y ->
  forall u, fib x u -> lt y u.
```

```
Theorem fib_squares: forall x x2, nat x -> times x x x2 ->
  (fib x x2 -> x = z \/ x = s z \/ x = s^12 z) /\
  (x = z \/ x = s z \/ x = s^12 z -> fib x x2).
... proof script here ...
... apply fib_square_above...
... proof script continued ...
```

abella publishing mode >> dispatch publish

bafyreihmsxveyp5so4neeah6rf35vuzj4urobaudy54...

dependencies: cid of fib_square_above(Abella) + other possible abella lemmas

conclusion: cid of fib_squares(Abella)

"I, some user, have indeed proved this theorem (`fib_square_above`), in Coq"

Lemma. For $n \in \mathbb{N}$, if $n \geq 13$, then $\mathrm{fib}(n) > n^2$.

ASSERTION-0

`fib_square_above`

`-->`

`fib_square_above`

"I, some user, have indeed proved this theorem (`fib_squares`), in Abella, depending on (`fib_square_above`)"

Theorem. For $n \in \mathbb{N}$, $\mathrm{fib}(n) = n^2$ if and only if $n \in \{0, 1, 12\}$, where $\mathrm{fib}(n)$ stands for the nth Fibonacci number defined as: $\mathrm{fib}(0) \triangleq 0$, $\mathrm{fib}(1) \triangleq 1$, and $\mathrm{fib}(n+2) \triangleq \mathrm{fib}(n+1) + \mathrm{fib}(n)$.

ASSERTION-1

The poor poet, Carl Spitzweg. Public domain, via Wikimedia Commons

DESIGN & IMPLEMENTATION

Linking the assertions: lookup

Who/what can I trust?

```
dependencies:[]
conclusion: cid of fib_square_above(Coq)
```

<K1, Coq>

```
dependencies:[]
conclusion: cid of fib_square_above(Coq)
```

<K1, Coq>

```
dependencies: cid of fib_square_above(Coq)
conclusion: cid of fib_square_above(Abella)
```

<K2, null>

```
dependencies:[]
conclusion: cid of fib_square_above(Coq)
```

<K1, Coq>

```
dependencies: cid of fib_square_above(Coq)
conclusion: cid of fib_square_above(Abella)
```

<K2, null>

```
dependencies: cid of fib_square_above(Abella) + ...
conclusion: cid of fib_squares(Abella)
```

<K2, Abella>

```
dependencies:[]
conclusion: cid of fib_square_above(Coq)
```

<K1, Coq>

```
dependencies: cid of fib_square_above(Coq)
conclusion: cid of fib_square_above(Abella)
```

<K2, null>

```
dependencies: cid of fib_square_above(Abella) + ...
conclusion: cid of fib_squares(Abella)
```

<K2, Abella>

Who/what can I trust?

```
dependencies:[]
conclusion: cid of fib_square_above(Coq)
```

```
<K1, Coq>
```

```
dependencies: cid of fib_square_above(Coq)
conclusion: cid of fib_square_above(Abella)
```

```
<K2, null>
```

```
dependencies: cid of fib_square_above(Abella) + ...
conclusion: cid of fib_squares(Abella)
```

```
<K2, Abella>
```

Who/what can I trust?

```
> lookup cid of fib_squares(Abella)
```

```
dependencies:[]
conclusion: cid of fib_square_above(Coq)
```

<K1, Coq>

```
dependencies: cid of fib_square_above(Coq)
conclusion: cid of fib_square_above(Abella)
```

<K2, null>

```
dependencies: cid of fib_square_above(Abella) + ...
conclusion: cid of fib_squares(Abella)
```

<K2, Abella>

Who/what can I trust?

```
> lookup cid of fib_squares(Abella)

 1. <K2, Abella> -- [cid of fib_square_above(Abella)]
```

dependencies:[]
conclusion: cid of fib_square_above(Coq) — <K1, Coq>

dependencies: cid of fib_square_above(Coq)
conclusion: cid of fib_square_above(Abella) — <K2, null>

dependencies: cid of fib_square_above(Abella) + ...
conclusion: cid of fib_squares(Abella) — <K2, Abella>

Who/what can I trust?

> lookup cid of fib_squares(Abella)

1. <K2, Abella> -- [cid of fib_square_above(Abella)]
2. <K2, Abella>, <K2, null> -- [cid of fib_square_above(Coq)]

```
dependencies:[]
conclusion: cid of fib_square_above(Coq)
```

```
<K1, Coq>
```

```
dependencies: cid of fib_square_above(Coq)
conclusion: cid of fib_square_above(Abella)
```

```
<K2, null>
```

```
dependencies: cid of fib_square_above(Abella) + ...
conclusion: cid of fib_squares(Abella)
```

```
<K2, Abella>
```

Who/what can I trust?

```
> lookup cid of fib_squares(Abella)
```

```
1. <K2, Abella> -- [cid of fib_square_above(Abella)]
2. <K2, Abella>, <K2, null>  -- [cid of fib_square_above(Coq)]
3. <K2, Abella>, <K2, null>, <K1, Coq> -- []
```

Ivan Ayvazovsky, Public domain, via Wikimedia Commons

REMARKS ON DAMF

The Distributed Assertion Management Framework

REMARKS ON DAMF

Natural existence of different communities, approaches, languages, logics, systems

Full translations not always needed for building and representing heterogeneous developments

Coherent information exchange in light of differences

Consensus, convergence possible, but not presumed

Cryptographic signatures for assertions (stamps)

Processing proofs for a reason, not always required

REMARKS ON DAMF

Information representation != trusting it

- no trust model is imposed, no truth notion is imposed

Arbitrary and tracked combination of assertions

- no logical consistency directly guaranteed
- stamps of agents on combinations and curations
- various forms of curations referring to common objects

Ivan Ayvazovsky, Public domain, via Wikimedia Commons

PROSPECTS

PROSPECTS

Design refinement through more questions

Annotations, where/how to store/represent proofs and evidence, metadata, proposed composition assertion, more on adapters

Improvement of lookup

Experimenting curation of libraries with human-readable names, granularity and compatibility of modes, etc

A visual, interactive DAMF browser

Graphical exploration of DAMF objects

Interactive compositions of assertions, etc

Ivan Ayvazovsky, Public domain, via Wikimedia Commons

The Distributed Assertion Management Framework - DAMF

A communication model that aims to enable communication between heterogeneous agents in a distributed environment, setting up the ground for emergent diverse structures

More on design, documentation of implementations, code, publications, and a full walkthrough of a heterogeneous development can be found at:

https://distributed-assertions.github.io/