Automated Proofs for Meshes

Matt Knepley

Computer Science and Engineering
University at Buffalo

Petsc/Firedrake 2026
Oxford, UK
June 2, 2026

University at Buffalo
Center for Hybrid Rocket
Exascale Simulation Technology

Never believe anything

until you run it.

Curry-Howard Isomorphism

Proof System $\Leftrightarrow$ Computational Model

Rocq

Calculus of Constructions

(**coquandhuet1988**)

Interfaces to regular code

What Tools Are Available?

Lean

Also Calculus of Constructions

Large library of mathematics

Outline

Plain English

	$\neg P$	$P \implies Q$
Hypothesis H	Prove P instead of False	Prove P instead of Q
Conclusion	Assume P, prove False	Assume P, prove Q

	$P \vee Q$	$P \wedge Q$
Hypothesis H	Prove using hyp. P then using hyp. Q	Replace with hyp. P and hyp. Q
Conclusion	Prove P or prove Q	Prove P, then prove Q

	$\forall x \in X, P(x)$	$\exists x \in X, P(x)$
Hypothesis H	Proves $P(y)$ for $y \in X$	Produce a witness $y \in X$ and the hyp. $P(y)$
Conclusion	Prove $P(x)$	Give a witness w

	$x = y$	$\mathcal{F}$
Hypothesis H	Substitute y for x Substitute x for y	False hyp. implies anything
Conclusion	A thing equals itself	

Rocq Syntax

	$\neg$	$\wedge$	$\vee$
Hypothesis `H`	`elim H`	`elim H` `destruct H as [H1 H2]`	`elim H` `destruct H as [H1 | H2]`
Conclusion	`intros H`	`split`	`left` or `right`

	$\Longrightarrow$	$\forall$	$\exists$
Hypothesis `H`	`apply H`	`elim H` `apply H`	`elim H` `destruct H as [x H1]`
Conclusion	`intros H`	`intros H`	`exists` w

	$=$	$\mathcal{F}$
Hypothesis `H`	`rewrite H` `rewrite <- H`	`elim H`
Conclusion	`reflexivity` `ring`	

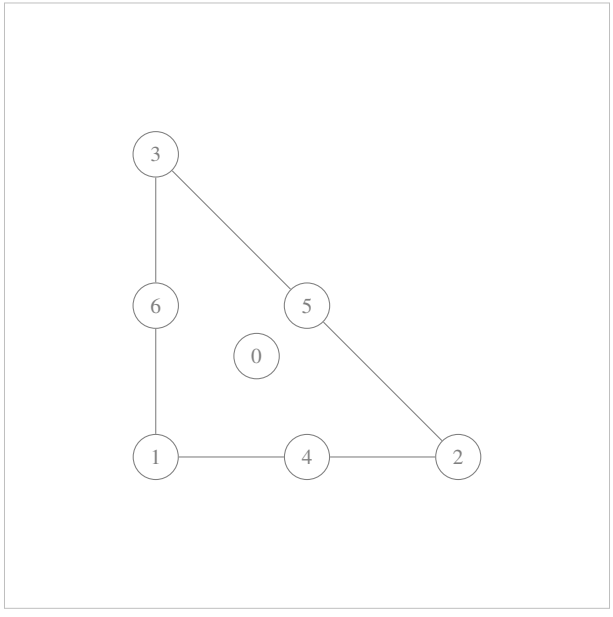
Outline

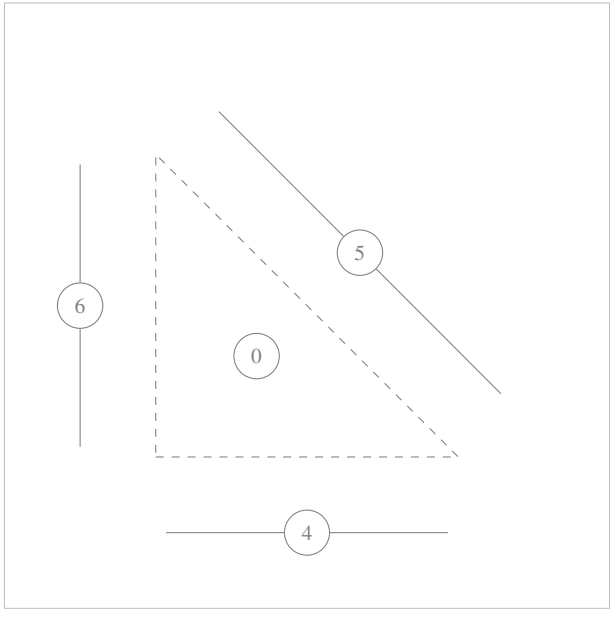
Mesh People

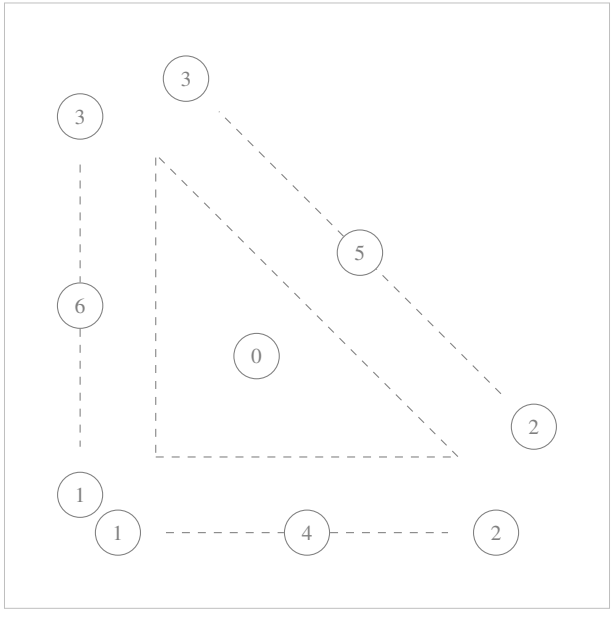
Toby Isaac

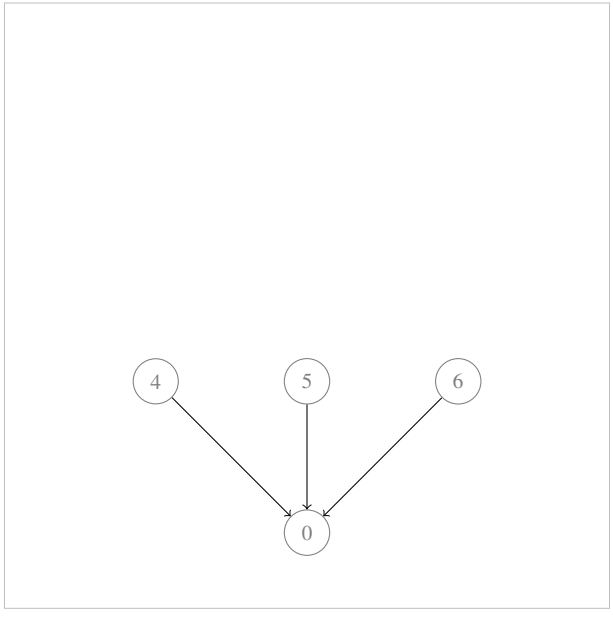
Vaclav Hapla

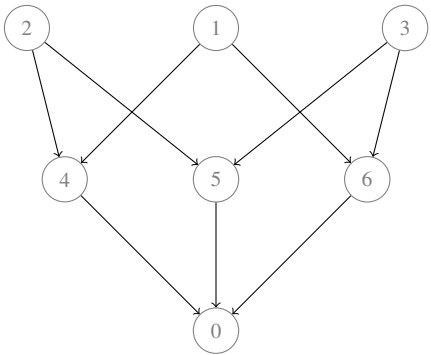
Stefano Zampini

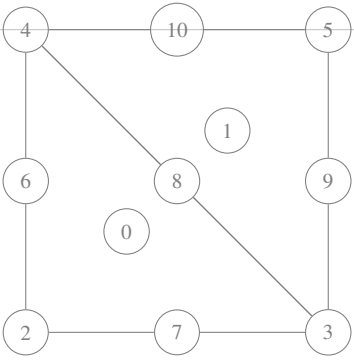

3
6
5
0
1
4
2

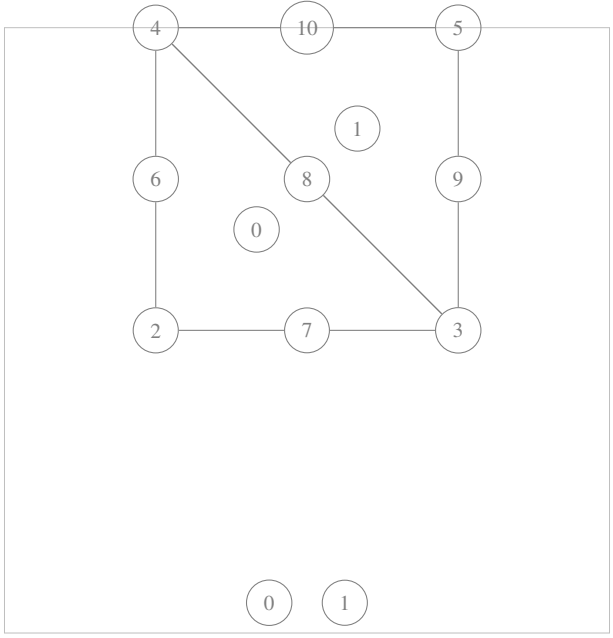

5
6
0
4

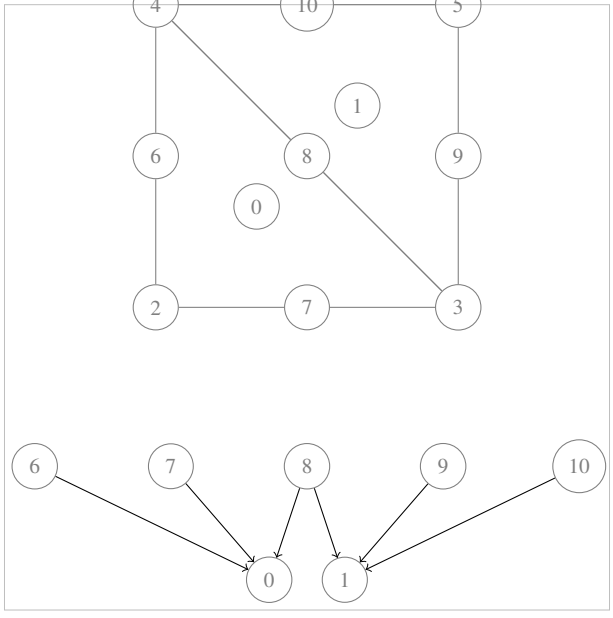

3
3
5
6
0
2
1
1
4
2

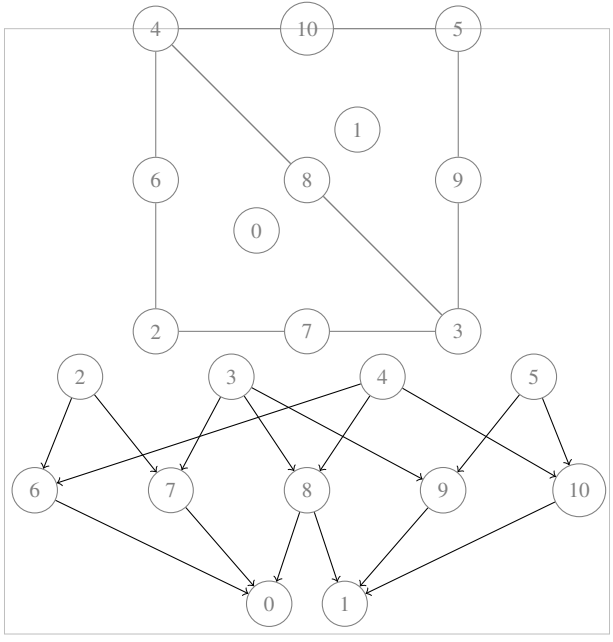

4
5
6
0

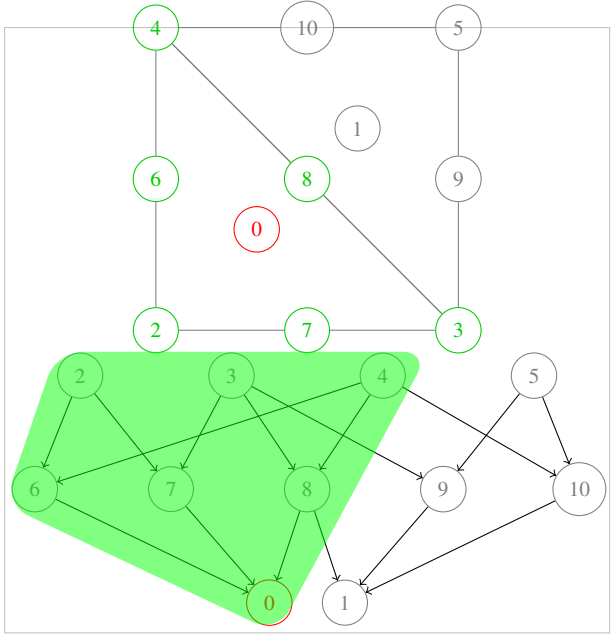

2
1
3
4
5
6
0

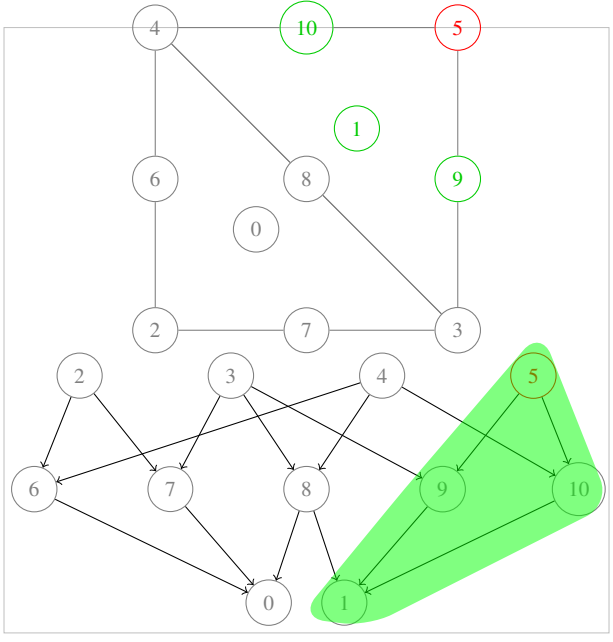

4
10
5
1
6
8
9
0
2
7
3


4
10
5
1
6
8
9
0
2
7
3
0
1


4
10
5
1
6
8
9
0
2
7
3
6
7
8
9
10
0
1


4
10
5
1
6
8
9
0
2
7
3
2
3
4
5
6
7
8
9
10
0
1


4
10
5
1
6
8
9
0
2
7
3
2
3
4
5
6
7
8
9
10
0
1


4
10
5
1
6
8
9
0
2
7
3
2
3
4
5
6
7
8
9
10
0
1

Plex in Rocq

```
Section Plex.
End Plex.
```

Plex in Rocq

```
Section Plex.
(* Types *)
Variable point : Type.
End Plex.
```

Plex in Rocq

```
Section Plex.
(* Types *)
Variable point : Type.
(* Mesh Operators *)
Variable cone: point -> point -> Prop.
Variable supp: point -> point -> Prop.
Inductive cl (p : point) : point -> Prop :=
  | cl_refl : cl p p
  | cl_step (q r : point) : cl p q -> cone q r -> cl p r.
Inductive st (p : point) : point -> Prop :=
  | st_refl : st p p
  | st_step (q r : point) : supp p q -> st q r -> st p r.
End Plex.
```

Plex Duality

```
Section Plex.
(* Symmetry *)
Axiom plex_dual : forall p q : point,
  (cone p q) <-> (supp q p).
(* Proofs *)
Theorem cl_st_dual : forall p q : point,
  (cl p q) <-> (st q p).
Theorem cl_trans : forall p q r : point,
  cl p q -> cl q r -> cl p r.
Theorem st_trans : forall p q r : point,
  st p q -> st q r -> st p r.
End Plex.
```

Hands-On Proofs!

Plex Duality

Proof Help

By inducting on `Hcl2`, we want to prove

$$\mathbb{cl}\ q\ r \longrightarrow \mathbb{cl}\ p\ r$$

so we choose

$$p \Longrightarrow q$$
$$P\ t \Longrightarrow \mathbb{cl}\ p\ t$$

Plex Duality

Proof Help

Base Case:

$$P\ p \Longrightarrow \mathbb{cl}\ p\ q$$

Inductive Step:

$$P\ q \Longrightarrow \mathbb{cl}\ p\ r$$
$$\mathbb{cone}\ q\ r \Longrightarrow \mathbb{cone}\ r\ s$$
$$P\ r \Longrightarrow \mathbb{cl}\ p\ s$$

Plex Transformations

```
Section Plex.
(* Transformation Operators *)
Variable child: point -> point -> Prop.
Variable parent: point -> point -> Prop.

(* Symmetry *)
Axiom plextr_dual : forall p a : point,
  (child p a) <-> (parent a p).
(* Surjectivity *)
Axiom plextr_surj : forall a : point,
  exists p : point, (child p a).
(* Locality *)
Axiom plextr_child_local : forall p a b : point,
  (child p a) -> (cone a b) ->
  exists q : point, (cl p q) /\ (child q b).
End Plex.
```

Hands-On Proofs!

Plex Transformations

```
Theorem plextr_local : forall p a b : point,
  (child p a) -> (cl a b) ->
  exists q : point, (cl p q) /\ (child q b).

Theorem plexnum_unique_local : forall p a b : point,
  (child p a) -> (st a b) ->
  (forall a p q : point, (parent a p) -> (parent a q) ->
    (p = q)) ->
  exists q : point, (st p q) /\ (child q b).

Theorem plexnum_interp_local : forall p a b : point,
  (child p a) -> (st a b) ->
  (forall a p q : point, (parent a p) -> (parent a q) ->
    (st p q)) ->
  exists q : point, (st p q) /\ (child q b).
```

Outline

Conclusions

- Proof verification is type checking
- Guides algorithm development
- Tools exist for C and OCaml
- Standard library being developed

Future

- Correctness proofs for parallel operations
- Modular integration is missing
- Formal specification of interfaces (MPI, . . .)
- Guide interface construction

References I