Computational Trinitarianism

Robert Harper

Programming Languages Mentoring Workshop

Rome, January 2013

The Holy Trinity of PL

The three manifestations of the divine:

- Type Theory = theory of computation
- Proof Theory = theory of proofs
- Category Theory = theory of mappings

Trinitarianism is the central organizing principle of PL research!

Classical Logic

- There are exactly 2 propositions, 0 and 1.
- Connectives are defined by *truth tables*.
- *Entailment*: A ≤ B iff A=1 implies B=1.
  - Reflexive, transitive (pre-order).
  - A true iff 1 ≤ A, A false iff A ≤ 0.
- The order structure is central, but often neglected!

Boolean Algebra

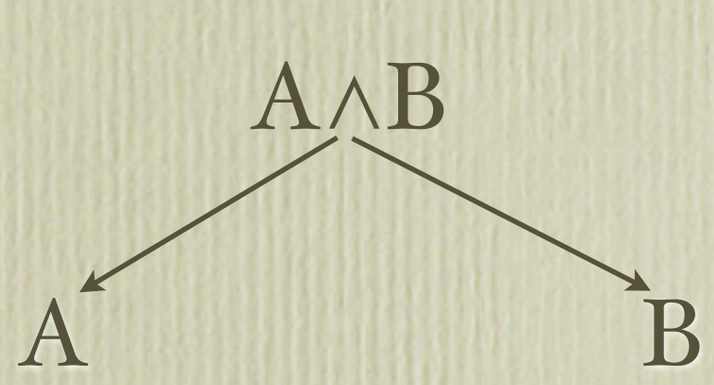
Boolean Algebra

- Conjunction defines the *meet* (aka *glb*):

- Disjunction defines the *join* (aka *lub*).
- 0 and 1 are least and greatest elements.

Boolean Algebra

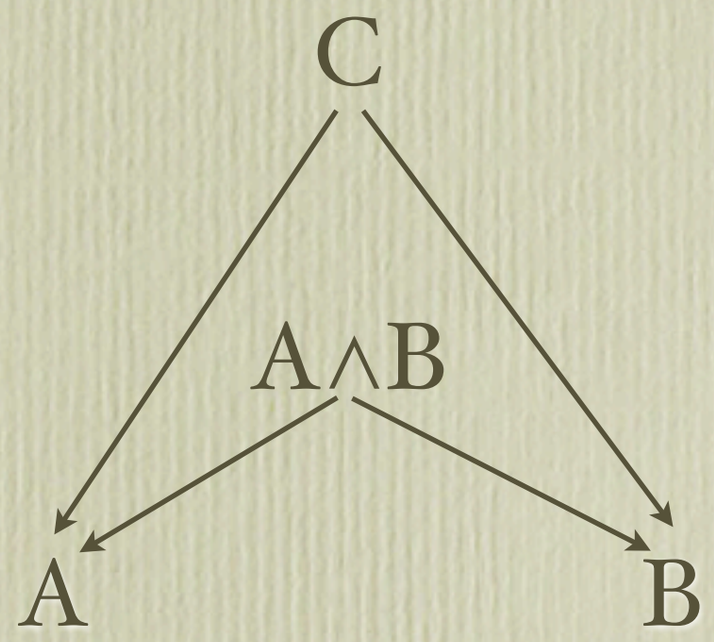
- Conjunction defines the *meet* (aka *glb*):



- Disjunction defines the *join* (aka *lub*).
- 0 and 1 are least and greatest elements.

Boolean Algebra

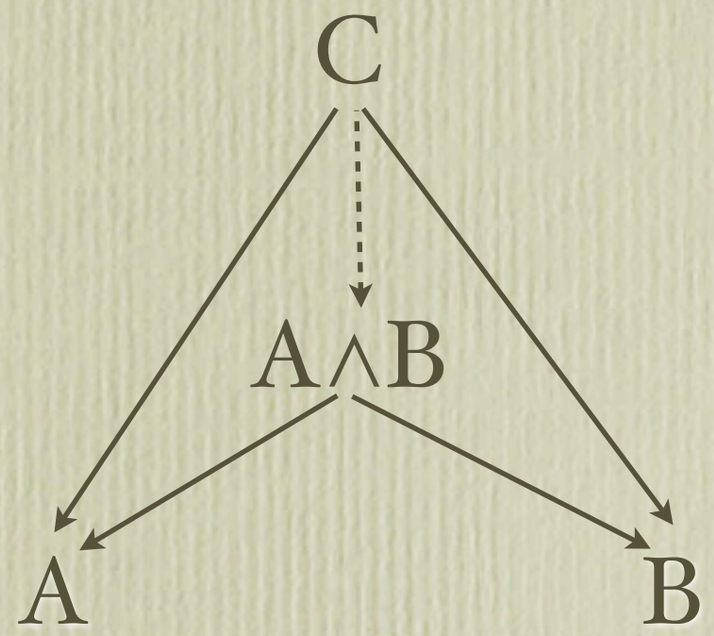
- Conjunction defines the *meet* (aka *glb*):



- Disjunction defines the *join* (aka *lub*).
- 0 and 1 are least and greatest elements.

Boolean Algebra

- Conjunction defines the *meet* (aka *glb*):



- Disjunction defines the *join* (aka *lub*).
- 0 and 1 are least and greatest elements.

Constructive Logic

- Classical logic gives no account of how *knowledge* is obtained or communicated.
  - A true iff there is a *proof* of A.
  - A false iff there is a *refutation* of A.
- But what is a proof?
- When are two proofs the same?

Logic as if People Matter

- Connectives are defined by *proof* conditions:
  - *Intro*: to prove A∧B true, it suffices to prove both A true and B true
  - *Elim*∧: from A∧B true we may deduce A true and B true.
- Similar rules govern the other connectives, by introduction and elimination rules.

Logical Entailment

- A_1 *true*, ..., A_n *true* $\vdash$ *B true* means there is a proof of B, given A_1, ..., A_n as axioms.
- *Reflexivity*: A true $\vdash$ A true (cf, $A \leq A$)
- *Weakening*:
  If $\Gamma \vdash$ B true, then Γ, A true $\vdash$ B true.
- *Transitivity*:
  If $\Gamma \vdash$ A true and Γ, A true $\vdash$ B true, then
  $\Gamma \vdash$ B true (cf, $A \leq B$ and $B \leq C$ implies $A \leq C$)

Logical Entailment

- Define connectives by *proof* conditions:
  - $A \wedge B$ true $\vdash$ A true
  - $A \wedge B$ true $\vdash$ B true
  - if C true $\vdash$ A true and C true $\vdash$ B true, then C true $\vdash A \wedge B$ true
- $\top$ true $\vdash$ A true is like $1 \leq A$ and A true $\vdash \bot$ true is like $A \leq 0$.

Proof Theory

- *Proof theory* is the study of proof terms.
  - Considered boring among logicians.
  - Of the essence for computer scientists!
- Two central concepts:
  - Dynamics: how to *execute* proofs.
  - Equivalence: when are two proofs the *same*?

The Language of Proof

- Enrich the rules of *truth* to obtain *rules of proof*.
  - $x_1 : A_1, \ldots, x_n : A_n \vdash M : A$ means
    M is a proof of A, given x_i a proof of A_i.
- Definition constitutes a *grammar of proof*.
  - Codify forms of argument.
- Proofs become *mathematical objects*.

The Language of Proof

- *Assumption*, or *variable*:
  $\Gamma, x{:}A \vdash x : A.$

- *Proliferation*:
  if $\Gamma \vdash N : B$, then $\Gamma, x{:}A \vdash N : B$.

- *Substitution*:
  if $\Gamma, x{:}A \vdash M : B$ and $\Gamma \vdash N : A$, then
  $\Gamma \vdash [N/x]M : B$

The Language of Proof

- Associate *proof terms* to each connective:
  - $x : A \wedge B \vdash \mathrm{fst}(x) : A$
  - $x : A \wedge B \vdash \mathrm{snd}(x) : B$
  - if $\Gamma \vdash M : A$ and $\Gamma \vdash N : B$, then $\Gamma \vdash \langle M, N \rangle : A \wedge B$
- A *formal grammar* of proof.

Gentzen's Principle

- Proofs have *computational content*, arising from the *inversion principle*:
  - fst ⟨M,N⟩ = M : A
  - snd ⟨M,N⟩ = N : A
- In general *elimination cancels introduction*, a *conservation principle* for proofs.

Brouwer's Principle

- All mathematical objects are *constructions*, or *computations*.
  - Geometric constructions.
    - Arithmetic computations: 2+2=4, etc.
- Proofs are *certain computations*, no different in kind from any other constructions.
  - Fundamentally *algorithmic* in nature.

Type Theory

Constructive Type Theory is the realization of Brouwer's Principle.

- Types classify computation.
  - Defined by introduction and elimination rules.

Propositions are types that classify proofs.

- Proofs are just computations.

Propositions and Types

	Proposition	*Type*	
truth	$\top$	1	*unit*
falsity	$\bot$	0	*void*
conjunction	$\wedge$	$\times$	*product*
disjunction	$\vee$	$+$	*sum*
implication	$\supset$	$\rightarrow$	*function*
		nat	*number*

Propositions and Types

	Proposition	*Type*	
negation	¬	cont	*continuation*
universal	∀	Π	*product*
existential	∃	Σ	*sum*
necessity	□	□	*mobility*
possibility	◇	◇	*locality*
laxity	OA	{A}	*monad*

Lawvere's Principle

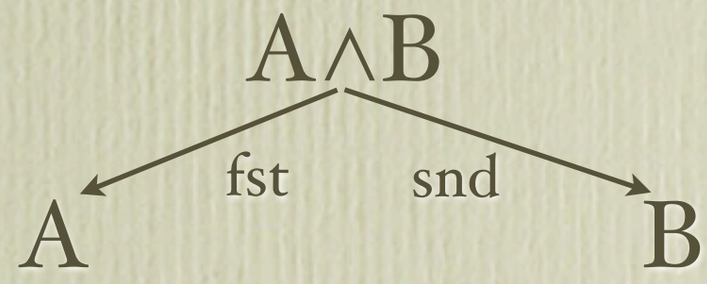
- Proofs can be thought of as *mappings*.
  - $\Gamma \vdash M : A$ as map $M : \Gamma \to A$.
    - Reflexivity: $\mathrm{id} : A \to A$
    - Transitivity: $N \circ M : A \to C$ if $M : A \to B$ and $N : B \to C$
- Generalize *preorders* to *categories*.

Lawvere's Dictum

- Maximality generalizes to universality:
  - $\langle M,N\rangle : \Gamma \to A \wedge B$ is "universal" among $M : \Gamma \to A$ and $N : \Gamma \to B$
- Pictorially:

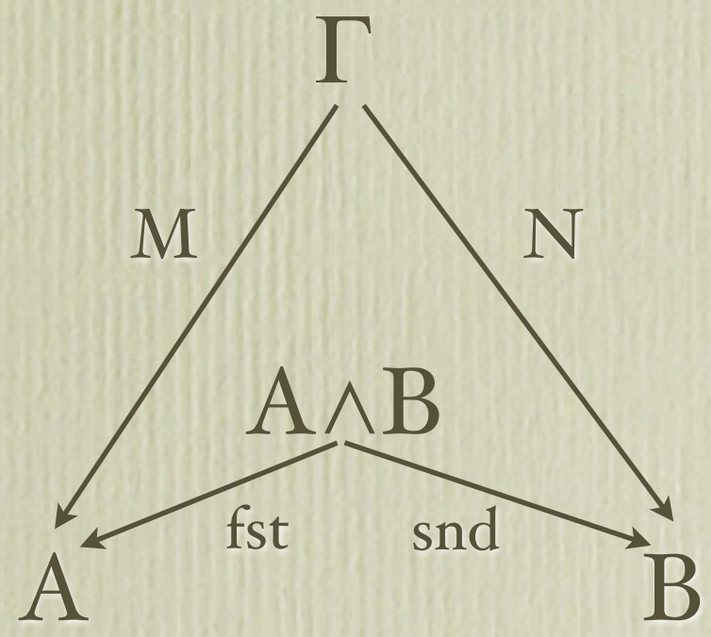
Lawvere's Dictum

- Maximality generalizes to universality:
  - $\langle M,N\rangle : \Gamma \to A \wedge B$ is "universal" among $M : \Gamma \to A$ and $N : \Gamma \to B$
- Pictorially:



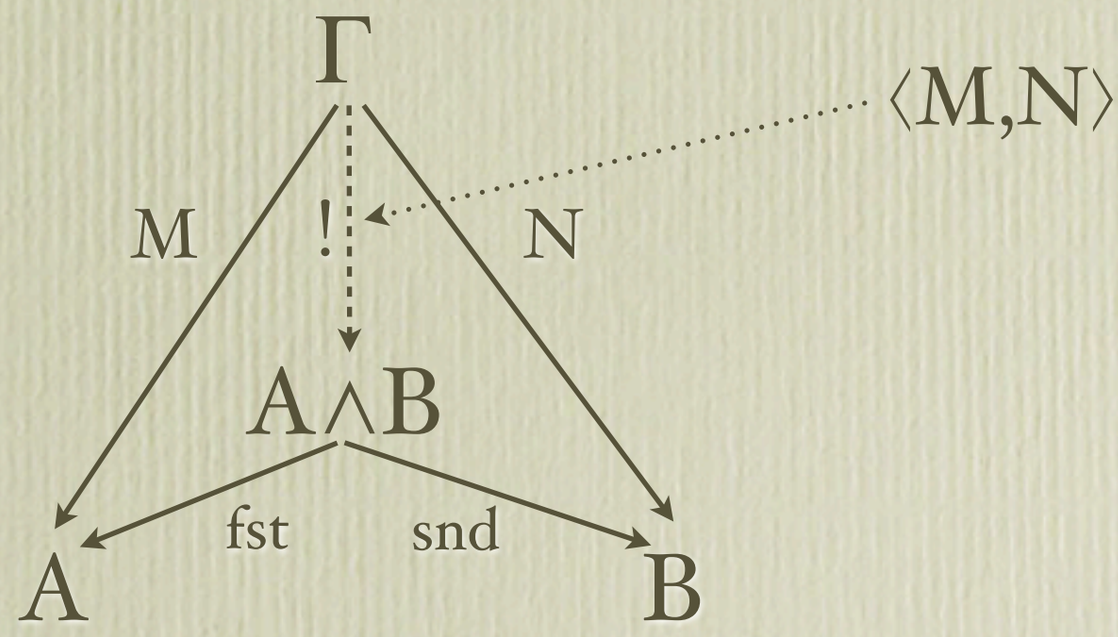
Lawvere's Dictum

- Maximality generalizes to universality:
  - $\langle M,N\rangle : \Gamma \to A \wedge B$ is "universal" among $M : \Gamma \to A$ and $N : \Gamma \to B$
- Pictorially:



Lawvere's Dictum

- Maximality generalizes to universality:
  - $\langle M,N\rangle : \Gamma \to A \wedge B$ is "universal" among $M : \Gamma \to A$ and $N : \Gamma \to B$
- Pictorially:



Lawvere's Principle

Universality expresses Gentzen's principles:

- $\mathrm{fst} \circ \langle M, N \rangle = M : A \wedge B \to A$
- $\mathrm{snd} \circ \langle M, N \rangle = N : A \wedge B \to B$
- $M = \langle \mathrm{fst} \circ M, \mathrm{snd} \circ N \rangle : \Gamma \to A \wedge B$

The last expresses an *equivalence of proofs*: any proof of a conjunction is equivalent to a pair.

What Is Equality?

- But when are two computations equal?
  - eg, $\lambda x.\ x+x =?= \lambda x.\ 2\times x$
    - different algorithms, same behavior
- When are two types equal?
  - eg, $N \to A \cong$ A stream are interchangeable

What Is Equality?

- Equality is a subtle business!
  - Definitional equality expresses computation, and requires no proof.
  - Propositional equality expresses equivalence, and requires proof.
- The type determines when two computations/proofs/objects are equal.

Propositional Equality

- Equivalences require *proof*.
  - $\alpha : M =_A N$ is proof that M and N are equal
    - There might be many such α's!
- Proofs of equivalence exhibit both *structure* and *computational content*.

Structure of Equivalence

- Proofs of equivalence form a *groupoid*:
  - Reflexive: $\mathrm{id} : M =_A M$
  - Transitive: $\beta \circ \alpha : M =_A P$ whenever $\alpha : N =_A P$ and $\beta : M =_A N$.
  - Symmetric: $\alpha^{-1} : N =_A M$ if $\alpha : M =_A N$
- Families and maps respect this structure.
  - $F(\alpha) : F(M) =_B F(N)$ if $F : A \to B$

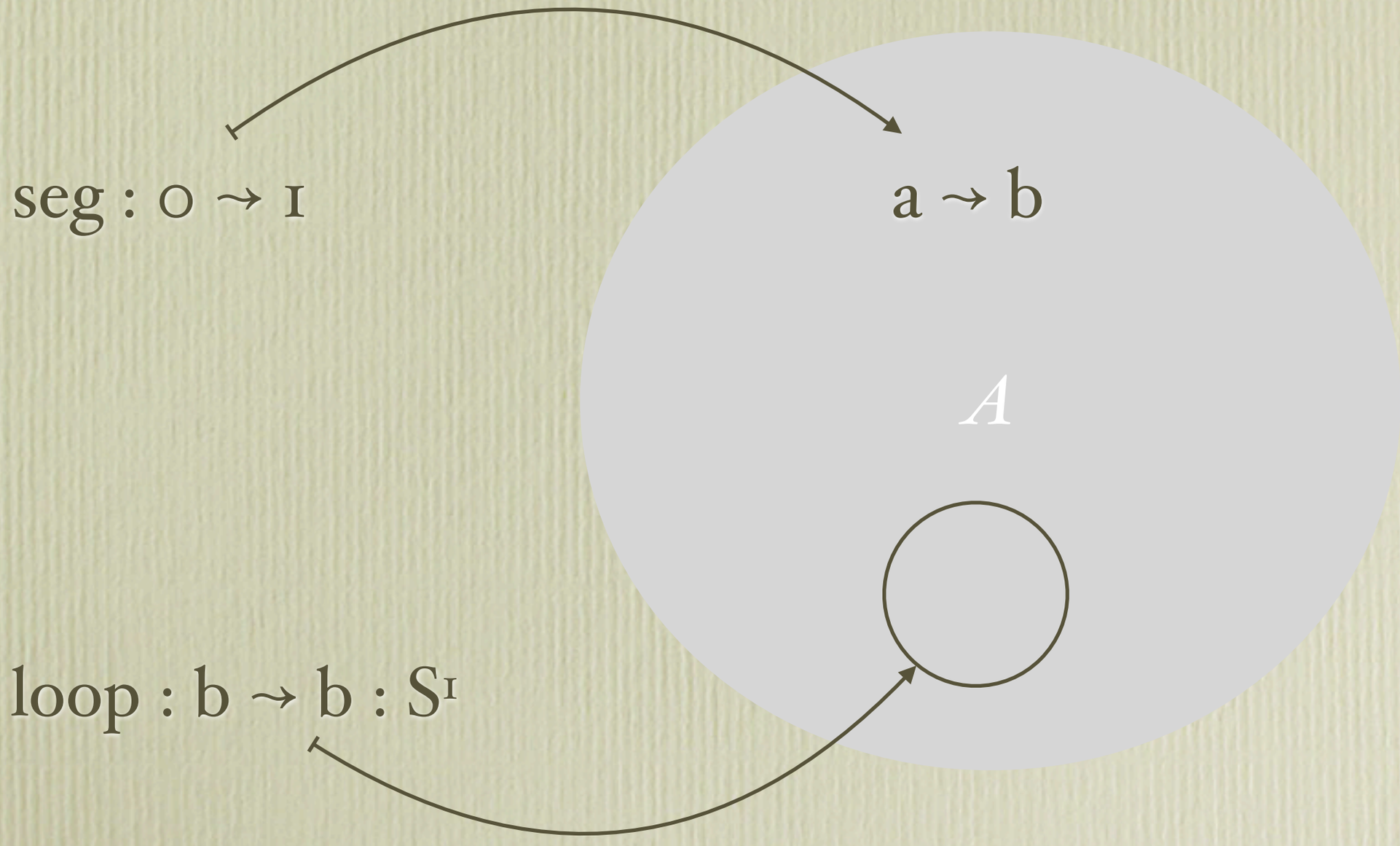
Equivalences as Paths

- Equivalences may be thought of as *paths* $\alpha : M \rightarrow_A N$ between M and N in A.
  - M may be *transported* to N within A
  - A may be *deformed* to bring M to N
- Allows *spaces* to be encoded as *higher inductive types* specifying *points* and *paths*.

Higher Inductive Types

- Interval, 1-sphere as higher inductive types:
  - I type; $0, 1 : I$; $seg : 0 \rightsquigarrow_I 1$
  - S type; $b : S$; $loop : b \rightsquigarrow_S b$
- Elimination by pattern-matching:
  - $p : I \to A$ given by $p\ 0 = a$, $p\ 1 = b$, and $p\ seg = \alpha : a \rightsquigarrow_A b$.
  - $c : S \to A$ given by $c\ b = a$, $c\ loop = \alpha : b \rightsquigarrow_A b$.

Higher Inductive Types



Voevodsky's Principle

Univalence, or *faithfulness*, states that any *equivalence* on a type may be regarded as a proof of *equality,* or a *path*, in that type.

- Bijective sets are *equal*.
- Bisimilar abstractions are *equal*.

Makes sense in a *constructive* setting, where proofs are computationally relevant.

- eg, $\alpha : A =_{Set} B$ sends elt(A) to elt(B) and vv.

Summary

Trinitarianism is the central organizing principle of PL research.

Exploiting the connections between PT, TT, and CT leads to new discoveries of fundamental importance.