A modal deconstruction of Löb induction

Daniel Gratzer

January 24, 2025

AARHUS UNIVERSITY

What are we trying to do?

Claim: recursive equations are the bedrock of denotational semantics for PLs.

See: untyped languages (λ-calculus), PCF, higher-order references, recursive types, etc.

What are we trying to do?

Claim: recursive equations are the bedrock of denotational semantics for PLs.

See: untyped languages (λ-calculus), PCF, higher-order references, recursive types, etc.

We can even prove it formally verify this claim in type theory!

$$\begin{aligned} &\mathsf{recEqsAreCool} : \mathsf{isCool}(\mathsf{RecursiveEquations}) \\ &\mathsf{recEqsAreCool} = \mathsf{fix}(\lambda \mathsf{prf} \to \mathsf{prf}) \end{aligned}$$

What are we trying to do?

Claim: recursive equations are the bedrock of denotational semantics for PLs.

See: untyped languages (λ-calculus), PCF, higher-order references, recursive types, etc.

We can even prove it formally verify this claim in type theory!

$$\begin{aligned}\mathsf{recEqsAreCool} &: \mathsf{isCool}(\mathsf{RecursiveEquations})\\ \mathsf{recEqsAreCool} &= \mathsf{fix}(\lambda \mathsf{prf} \to \mathsf{prf})\end{aligned}$$

recursive equations & type theory do not mix.

Guarded type theory: the basic components

A proposed answer: guarded type theory!

- fix is safe as long as we "do work" before recurring
- Use a unary type constructor (modality) to crystallize this discipline.

We arrive at the basic components of guarded recursion:

$$\blacktriangleright : \mathcal{U} \to \mathcal{U} \qquad\qquad \mathsf{loeb} : (A : \mathcal{U}) \to (\blacktriangleright A \to A) \to A$$

Guarded type theory: the basic components

A proposed answer: guarded type theory!

- fix is safe as long as we "do work" before recurring
- Use a unary type constructor (modality) to crystallize this discipline.

We arrive at the basic components of guarded recursion:

$$\blacktriangleright : \mathcal{U} \to \mathcal{U} \qquad\qquad \mathsf{loeb} : (A : \mathcal{U}) \to (\blacktriangleright A \to A) \to A$$

$$\mathsf{recEqsAreCool} : \mathsf{isCool}(\mathsf{RecursiveEquations})$$

$$\mathsf{recEqsAreCool} = \mathsf{loeb}(\lambda \mathsf{prf} \to \textcolor{red}{\mathsf{prf}})$$

Guarded type theory: the basic components

A proposed answer: guarded type theory!

- fix is safe as long as we "do work" before recurring
- Use a unary type constructor (modality) to crystallize this discipline.

We arrive at the basic components of guarded recursion:

$$\blacktriangleright : \mathcal{U} \to \mathcal{U} \qquad\qquad \mathsf{loeb} : (A : \mathcal{U}) \to (\blacktriangleright A \to A) \to A$$

$$\mathsf{recEqsAreCool} : \mathsf{isCool}(\mathsf{RecursiveEquations})$$

$$\mathsf{recEqsAreCool} = \mathsf{loeb}(\lambda \mathsf{prf} \to \mathsf{prf})$$

Lots of ways to structure this [Nak00; AM13; BM13; Møg14; BGM17; GB22; KMV22]

Now the semantics of ▶

Guarded recursion $\sim$ proof-relevant step-indexing (via $\mathbf{PSh}(\omega)$)

$$\begin{array}{ccc} & & \cdots \\ & & \downarrow \\ & & X_2 \\ & & \downarrow \\ [\![X]\!] & = & X_1 \\ & & \downarrow \\ & & X_0 \end{array} \qquad\qquad \begin{array}{ccc} & & \cdots \\ & & \downarrow \\ & & X_1 \\ & & \downarrow \\ [\![\blacktriangleright X]\!] & = & X_0 \\ & & \downarrow \\ & & \{\star\} \end{array}$$

Types $\sim$ Time-indexed sets

Terms $\sim$ Time-preserving maps

Now the semantics of ▶

Guarded recursion $\sim$ proof-relevant step-indexing (via $\mathbf{PSh}(\omega)$)

$$
\begin{array}{ccc}
 & & \cdots \\
 & & \downarrow \\
 & & X_2 \\
 & & \downarrow \\
[\![X]\!] & = & X_1 \\
 & & \downarrow \\
 & & X_0
\end{array}
\qquad\qquad
\begin{array}{ccc}
 & & \cdots \\
 & & \downarrow \\
 & & X_1 \\
 & & \downarrow \\
[\![\blacktriangleright X]\!] & = & X_0 \\
 & & \downarrow \\
 & & \{\star\}
\end{array}
$$

Types $\sim$ Time-indexed sets

Terms $\sim$ Time-preserving maps

Are we done?

A few things are left to nail down:

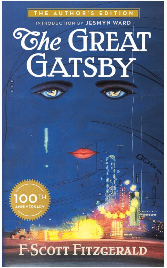
1. $\blacktriangleright$ is part of a whole family of modal operators, what about them?
2. $\blacktriangleright$/loeb are sound, but what about canonicity, normalization, etc.

The challenge: how do we have a full dependent type theory with (1) and (2).

Contributions: Gatsby

Turns out, we can reduce this to a question of modalities:

- We show that adding a new principle *about modalities* allows us to derive Löb.
- Crucially rely on univalence/HoTT in a few places...
- All told, obtain a new framework for guarded recursion: Gatsby.



Definitely satisfies (1) and have good evidence that (2) is also true.

Guarded type theory: modalities

One modality $\blacktriangleright$ (also written $\langle \ell \mid - \rangle$ for uniformity) is good, surely more is better!

$$
\begin{tikzcd}
t \arrow[loop left, "{\ell, e}"] \arrow[rr, bend left, "\epsilon_0"] \arrow[rr, bend right, "\gamma" description] & & s \arrow[ll, "\delta" description] \arrow[loop right, "\top"]
\end{tikzcd}
$$

$$
\begin{aligned}
e \circ \ell &= \mathrm{id} & \epsilon_0 \circ \ell &= \top \\
\gamma \circ \delta &= \mathrm{id} & \mathrm{id} &\leq \delta \circ \gamma \\
\dots
\end{aligned}
$$

- Two classes of types: t for those which are guarded, s for standard types
- More modalities $\sim$ more control over "how fast" terms produce answers.
- Ex. $\langle e \mid A \rangle$ is "an A which has already done work" so $\langle e \mid \langle \ell \mid A \rangle \rangle \simeq A$.

Guarded type theory: modalities

One modality $\blacktriangleright$ (also written $\langle \ell \mid - \rangle$ for uniformity) is good, surely more is better!

$$\begin{tikzcd} t \arrow[loop left, "{\ell, e}"] \arrow[rr, bend left, "\epsilon_0"] \arrow[rr, bend right, "\gamma" description] & & s \arrow[ll, "\delta" description] \arrow[loop right, "\top"] \end{tikzcd}$$

$$e \circ \ell = \mathrm{id} \qquad \epsilon_0 \circ \ell = \top$$
$$\gamma \circ \delta = \mathrm{id} \qquad \mathrm{id} \leq \delta \circ \gamma$$
$$\ldots$$

- Two classes of types: t for those which are guarded, s for standard types
- More modalities $\sim$ more control over "how fast" terms produce answers.
- Ex. $\langle e \mid A \rangle$ is "an A which has already done work" so $\langle e \mid \langle \ell \mid A \rangle \rangle \simeq A$.

Guarded type theory: modalities

One modality $\blacktriangleright$ (also written $\langle \ell \mid - \rangle$ for uniformity) is good, surely more is better!

$$
\begin{tikzcd}
t \arrow[loop left, "{\ell, e}"] \arrow[rr, bend left, "\epsilon_0"] \arrow[rr, bend right, "\gamma"'] & & s \arrow[ll, "\delta" description] \arrow[loop right, "\top"]
\end{tikzcd}
\qquad
\begin{array}{ll}
e \circ \ell = \mathrm{id} & \epsilon_0 \circ \ell = \top \\
\gamma \circ \delta = \mathrm{id} & \mathrm{id} \leq \delta \circ \gamma \\
\ldots &
\end{array}
$$

- Two classes of types: t for those which are guarded, s for standard types
- More modalities $\sim$ more control over "how fast" terms produce answers.
- Ex. $\langle e \mid A \rangle$ is "an A which has already done work" so $\langle e \mid \langle \ell \mid A \rangle \rangle \simeq A$.

Guarded type theory: modalities

One modality $\blacktriangleright$ (also written $\langle \ell \mid - \rangle$ for uniformity) is good, surely more is better!

$$
\begin{tikzcd}
t \arrow[loop left, "{\ell, e}"] \arrow[rr, bend left, "\epsilon_0"] \arrow[rr, bend right, "\gamma"'] & & s \arrow[ll, "\delta"] \arrow[loop right, "\top"]
\end{tikzcd}
$$

$$
\begin{array}{ll}
e \circ \ell = \mathsf{id} & \epsilon_0 \circ \ell = \top \\
\gamma \circ \delta = \mathsf{id} & \mathsf{id} \leq \delta \circ \gamma \\
\dots &
\end{array}
$$

- Two classes of types: t for those which are guarded, s for standard types
- More modalities $\sim$ more control over "how fast" terms produce answers.
- Ex. $\langle e \mid A \rangle$ is "an A which has already done work" so $\langle e \mid \langle \ell \mid A \rangle \rangle \simeq A$.

Guarded type theory: modalities

One modality $\blacktriangleright$ (also written $\langle \ell \mid - \rangle$ for uniformity) is good, surely more is better!

$$
\begin{tikzcd}
t \arrow[loop left, "\ell, e"] \arrow[rr, bend left, "\epsilon_0"] \arrow[rr, bend right, "\gamma"'] & & s \arrow[loop right, "\top"] \arrow[ll, "\delta"']
\end{tikzcd}
\qquad
\begin{array}{ll}
e \circ \ell = \mathrm{id} & \epsilon_0 \circ \ell = \top \\
\gamma \circ \delta = \mathrm{id} & \mathrm{id} \leq \delta \circ \gamma \\
\dots &
\end{array}
$$

- Two classes of types: t for those which are guarded, s for standard types
- More modalities $\sim$ more control over "how fast" terms produce answers.
- Ex. $\langle e \mid A \rangle$ is "an A which has already done work" so $\langle e \mid \langle \ell \mid A \rangle \rangle \simeq A$.

Guarded type theory: modalities

One modality $\blacktriangleright$ (also written $\langle \ell \mid - \rangle$ for uniformity) is good, surely more is better!

$$
\begin{tikzcd}
t \arrow[loop left, "{\ell, e}"] \arrow[rr, bend left, "\epsilon_0"] \arrow[rr, bend right, "\gamma" description] & & s \arrow[ll, "\delta" description] \arrow[loop right, "\top"]
\end{tikzcd}
$$

$$
\begin{array}{ll}
e \circ \ell = \mathrm{id} & \epsilon_0 \circ \ell = \top \\
\gamma \circ \delta = \mathrm{id} & \mathrm{id} \leq \delta \circ \gamma \\
\ldots &
\end{array}
$$

- Two classes of types: t for those which are guarded, s for standard types
- More modalities $\sim$ more control over "how fast" terms produce answers.
- Ex. $\langle e \mid A \rangle$ is "an A which has already done work" so $\langle e \mid \langle \ell \mid A \rangle \rangle \simeq A$.

Guarded type theory: modalities

One modality $\blacktriangleright$ (also written $\langle \ell \mid - \rangle$ for uniformity) is good, surely more is better!

$$\begin{tikzcd}
t \arrow[loop left, "{\ell, e}"] \arrow[rr, bend left, "\epsilon_0"] \arrow[rr, bend right, "\gamma"'] & & s \arrow[ll, "\delta" description] \arrow[loop right, "\top"]
\end{tikzcd}$$

$$\begin{array}{ll}
e \circ \ell = \mathsf{id} & \epsilon_0 \circ \ell = \top \\
\gamma \circ \delta = \mathsf{id} & \mathsf{id} \leq \delta \circ \gamma \\
\ldots &
\end{array}$$

- Two classes of types: t for those which are guarded, s for standard types
- More modalities $\sim$ more control over "how fast" terms produce answers.
- Ex. $\langle e \mid A \rangle$ is "an A which has already done work" so $\langle e \mid \langle \ell \mid A \rangle \rangle \simeq A$.

Guarded type theory: modalities

One modality $\blacktriangleright$ (also written $\langle \ell \mid - \rangle$ for uniformity) is good, surely more is better!

$$\begin{tikzcd}
t \arrow[loop left, "{\ell, e}"] \arrow[rr, bend left, "\epsilon_0"] \arrow[rr, bend right, "\gamma" description] & & s \arrow[ll, "\delta" description] \arrow[loop right, "\top"]
\end{tikzcd}$$

$$\begin{aligned}
e \circ \ell &= \mathsf{id} & \epsilon_0 \circ \ell &= \top \\
\gamma \circ \delta &= \mathsf{id} & \mathsf{id} &\leq \delta \circ \gamma \\
&\dots
\end{aligned}$$

- Two classes of types: t for those which are guarded, s for standard types
- More modalities $\sim$ more control over “how fast” terms produce answers.
- Ex. $\langle e \mid A \rangle$ is “an A which has already done work” so $\langle e \mid \langle \ell \mid A \rangle \rangle \simeq A$.

Relatively new: adding modalities & univalence is cheap [Gra+20; Aag+22].

Guarded type theory: Löb induction

Theorem

No matter the combination of modalities, we cannot define $\mathsf{loeb} : (\langle \ell \mid A \rangle \to A) \to A$.

Proof.

Nothing allows us to internally prove $\langle \ell \mid A \rangle \neq A$. □

Pictured: Gatsby longing for Löb induction.

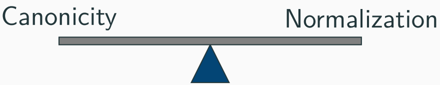
Löb induction in type theory

Theorem (Gratzer and Birkedal [GB22])

If loeb *computes then type-checking is undecidable.*

Lemma

If loeb *never computes then canonicity fails.*



- Clocked type theory [BGM19; KMV22], let loeb compute only sometimes...
- Adding loeb is hard, I would prefer not to.

Big idea: move the goal posts

Let's play a new game:

$$💣 = \left\| \sum_{n:\mathbf{Nat}} \blacktriangleright^{n} \bot \right\|$$

Big idea: move the goal posts

Let's play a new game:

$$💣 = \left\|\sum_{n:\mathbf{Nat}} \blacktriangleright^{n}\bot\right\|$$

$\|-\|$ is propositional truncation, so read $\exists n : \mathbf{Nat}.\, \blacktriangleright^{n}\bot$

Holds in $\mathbf{PSh}(\omega)$; choice of n doesn't have to agree at each step!

Big idea: move the goal posts

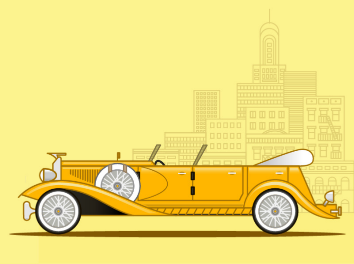
Let's play a new game:

$$💣 = \left\| \sum_{n:\mathbf{Nat}} \blacktriangleright^{n} \bot \right\|$$

Key Idea

💣 is a *doomsday proposition*; hypothesizes everything will collapse... eventually.

Pictured: Well-known symbolism for 💣.



From existential dread to Löb induction

Theorem

If 💣 *then every operation* $f : \langle \ell \mid A \rangle \to A$ *has a* ***unique*** *guarded fixed-point:*

$$💣 \to \mathsf{isContr}\left(\sum_{a:A} \text{“}a \text{ is a guarded fixed-point of } f\text{”}\right)$$

Proof Sketch.

The goal is an (h-)proposition, so ignore $\|-\|$ and use induction on $n : \mathbf{Nat}$. $\square$

Univalence Alert

Relies on HoTT's more *semantic* notion of proposition/truncation.

Goal posts shifted... Now what to do with 💣?

If 💣 were true, we'd be done... but this breaks decidable type-checking [GB22]

Key Idea

It suffices to find a supply of (very cynical) types which *believe* 💣 to be true.

Definition

A is accessible if the canonical map $A \to ($💣$\to A)$ is an equivalence.

Goal posts shifted... Now what to do with 💣?

If 💣 were true, we'd be done... but this breaks decidable type-checking [GB22]

Key Idea

It suffices to find a supply of (very cynical) types which *believe* 💣 to be true.

Definition

A is accessible if the canonical map $A \to (💣 \to A)$ is an equivalence.

Theorem

If A is accessible then $\mathrm{loeb} : (\blacktriangleright A \to A) \to A$

Not every type is accessible (for instance, not 💣!) but a lot of them are...

Accessible types

Theorem (Rijke, Shulman, and Spitters [RSS20])

Accessible types form a reflective subuniverse closed under Π, Σ, $=$, $\mathbf{1}$, *and* $\mathcal{U}_{\text{💣}}$.

Univalence Alert

Crucial use of univalence to show that $\mathcal{U}_{\text{💣}} = \sum_{A:\mathcal{U}} \mathrm{isAcc}(A)$ is accessible.

This is all well and good, but we want some non-trivial base types!

Enter $\top$

To move further, we turn back to our new modality: $\top : s \longrightarrow s$:

Key Idea

Add one new rule to force $\langle \top \mid A \rangle \simeq \mathbf{Unit}$.

Immediate consequence: $\langle \epsilon_0 \mid \langle \ell \mid A \rangle \rangle \simeq \langle \top \mid A \rangle$

Enter $\top$

To move further, we turn back to our new modality: $\top : s \longrightarrow s$:

Key Idea

Add one new rule to force $\langle \top \mid A \rangle \simeq \textbf{Unit}$.

Immediate consequence: $\langle \epsilon_0 \mid \langle \ell \mid A \rangle \rangle \simeq \langle \top \mid A \rangle$

$$\frac{\text{For the } \Gamma \text{ and } \vdash \text{ rules forcing } \langle \top \mid A \rangle \simeq \textbf{Unit}}{\text{See the paper!}}$$

The world's hardest working unit type

Remarkably, $\top$'s existence yields more accessible types!

Theorem

- *If* $\langle \ell \mid \mathrm{isAcc}(A) \rangle$ *then* $\langle \ell \mid A \rangle$ *is accessible.*
- *For any s type (i.e., not guarded)* A*,* $\langle \delta \mid A \rangle$ *is accessible.*

Corollary

Being accessible is closed under $\mathbf{0}$, $+$, $\mathbf{Bool}$, *and* $\mathbf{Nat}$.

Checking the scoreboard... that's pretty much all the type-formers!

The payoff

Since so many types are accessible, we can begin work exclusively with accessible types.

Theorem

Standard guarded type theory ($\square, \blacktriangleright$ etc.) has a model in accessible types.

- Since e.g., **Nat** is accessible, we can use loeb to compute concrete numbers!
- It also means that users of Gatsby never actually have to talk about 💣

A few case studies

Extra modalities & standard v guarded type separation give new opportunities:

- We define a logical relation for general references using loeb for type safety
- We construct non-guarded coinductive types using guarded recursion internally.
- Both of these prove results about non-guarded objects using guarded recursion!

Via a semantic model, an adequate proof strategy: cubical version of $\mathbf{PSh}(\omega)$.

Is Gatsby great?

We show that adding a new principle *about modalities* allows us to derive Löb.

- Crucially rely on univalence/HoTT in a few places...
- All told, obtain a new framework for guarded recursion.

Definitely satisfies (1) and have good evidence that (2) is also true.

A few words on the proof

Theorem

- *If* $\langle \ell \mid \mathrm{isAcc}(A) \rangle$ *then* $\langle \ell \mid A \rangle$ *is accessible.*
- *For any s type (i.e., not guarded)* A*,* $\langle \delta \mid A \rangle$ *is accessible.*

The proof amounts to two crucial facts:

- $\langle e \mid \text{💣} \rangle \simeq \text{💣}$
- $\langle \epsilon_0 \mid \text{💣} \rangle \simeq \mathbf{1}$.

Proven formal modal shuffling & the fact that $\langle \top \mid A \rangle = \mathbf{1}$.

Key Idea

Novel rule means $\langle \top \mid A \rangle \neq A$ and equations between modalities then ripple out.

References

[Aag+22] Frederik Lerbjerg Aagaard, Magnus Baunsgaard Kristensen, Daniel Gratzer, and Lars Birkedal. *Unifying cubical and multimodal type theory*. 2022. arXiv: 2203.13000 [cs.LO] (cit. on pp. 12–19).

[AM13] Robert Atkey and Conor McBride. "Productive Coprogramming with Guarded Recursion". In: *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming*. ICFP '13. Association for Computing Machinery, 2013, pp. 197–208. DOI: 10.1145/2500365.2500597. URL: https://doi.org/10.1145/2500365.2500597 (cit. on pp. 5–7).

[BGM17] Patrick Bahr, Hans Bugge Grathwohl, and Rasmus Ejlers Møgelberg. "The clocks are ticking: No more delays!" In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2017. DOI: 10.1109/LICS.2017.8005097. URL: http://www.itu.dk/people/mogel/papers/lics2017.pdf (cit. on pp. 5–7).

[BGM19] Patrick Bahr, Christian Uldal Graulund, and Rasmus Ejlers Møgelberg. "Simply RaTT: A Fitch-style Modal Calculus for Reactive Programming Without Space Leaks". In: *Proc. ACM Program. Lang.* 3 (ICFP 2019), 109:1–109:27. ISSN: 2475-1421. DOI: 10.1145/3341713 (cit. on p. 21).

[BM13] Lars Birkedal and Rasmus Ejlers Møgelberg. "Intensional Type Theory with Guarded Recursive Types qua Fixed Points on Universes". In: *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '13. USA: IEEE Computer Society, 2013, pp. 213–222. ISBN: 9780769550206. DOI: 10.1109/LICS.2013.27 (cit. on pp. 5–7).

[GB22] Daniel Gratzer and Lars Birkedal. "A Stratified Approach to Löb Induction". In: *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*. Ed. by Amy Felty. Vol. 228. Leibniz International Proceedings in Informatics (LIPIcs). Saarbrücken, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2022. DOI: 10.4230/LIPIcs.FSCD.2022.3. URL: https://jozefg.github.io/papers/a-stratified-approach-to-lob-induction.pdf (cit. on pp. 5–7, 21, 26, 27).

[Gra+20] Daniel Gratzer, G.A. Kavvos, Andreas Nuyts, and Lars Birkedal. "Multimodal Dependent Type Theory". In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '20. ACM, 2020. DOI: 10.1145/3373718.3394736 (cit. on pp. 12–19).

[KMV22] Magnus Baunsgaard Kristensen, Rasmus Ejlers Møgelberg, and Andrea Vezzosi. "Greatest HITs: Higher inductive types in coinductive definitions via induction under clocks". In: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2022. DOI: 10.1145/3531130.3533359 (cit. on pp. 5–7, 21).

[Møg14] Rasmus Ejlers Møgelberg. "A Type Theory for Productive Coprogramming via Guarded Recursion". In: *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. CSL-LICS '14. 2014. DOI: 10.1145/2603088.2603132 (cit. on pp. 5–7).

[Nak00] H. Nakano. "A modality for recursion". In: *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No.99CB36332)*. IEEE Computer Society, 2000, pp. 255–266 (cit. on pp. 5–7).

[RSS20] Egbert Rijke, Michael Shulman, and Bas Spitters. "Modalities in homotopy type theory". In: *Logical Methods in Computer Science* 16.1 (2020). eprint: 1706.07526 (cit. on p. 28).