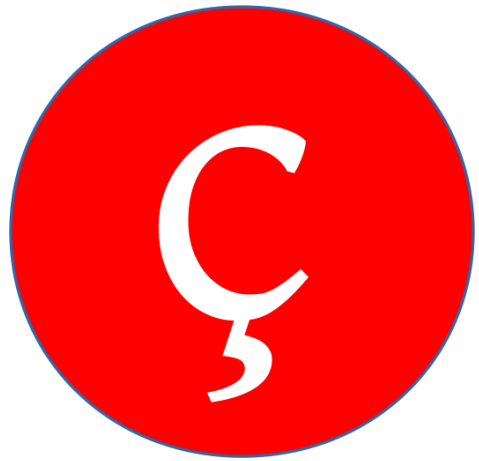
Introduction to Programming and Proving in Cedille



Chris Jenkins, Colin McDonald, Aaron Stump
Computer Science
The University of Iowa
Iowa City, Iowa

Plan for the tutorial

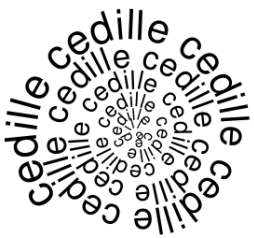
?

⊢ *CeDilLE*

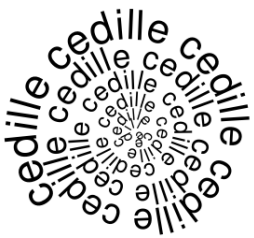
`cedille`

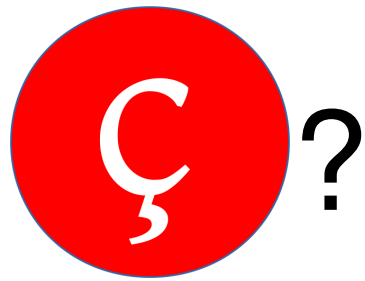
⇝ `cedille`$_{\texttt{core}}$

`c d ll`



Plan for the tutorial

Ç?	Motivation and background for Cedille
$\vdash$ *CeDilLE*	Syntax and semantics
`cedille`	Tooling: emacs frontend $\leftrightarrow$ backend
$\rightsquigarrow$ `cedille`$_{\texttt{core}}$	Elaboration to Cedille Core
`c d ll`	Spine-local type inference
	Future directions



Motivation and background for Cedille

A little history

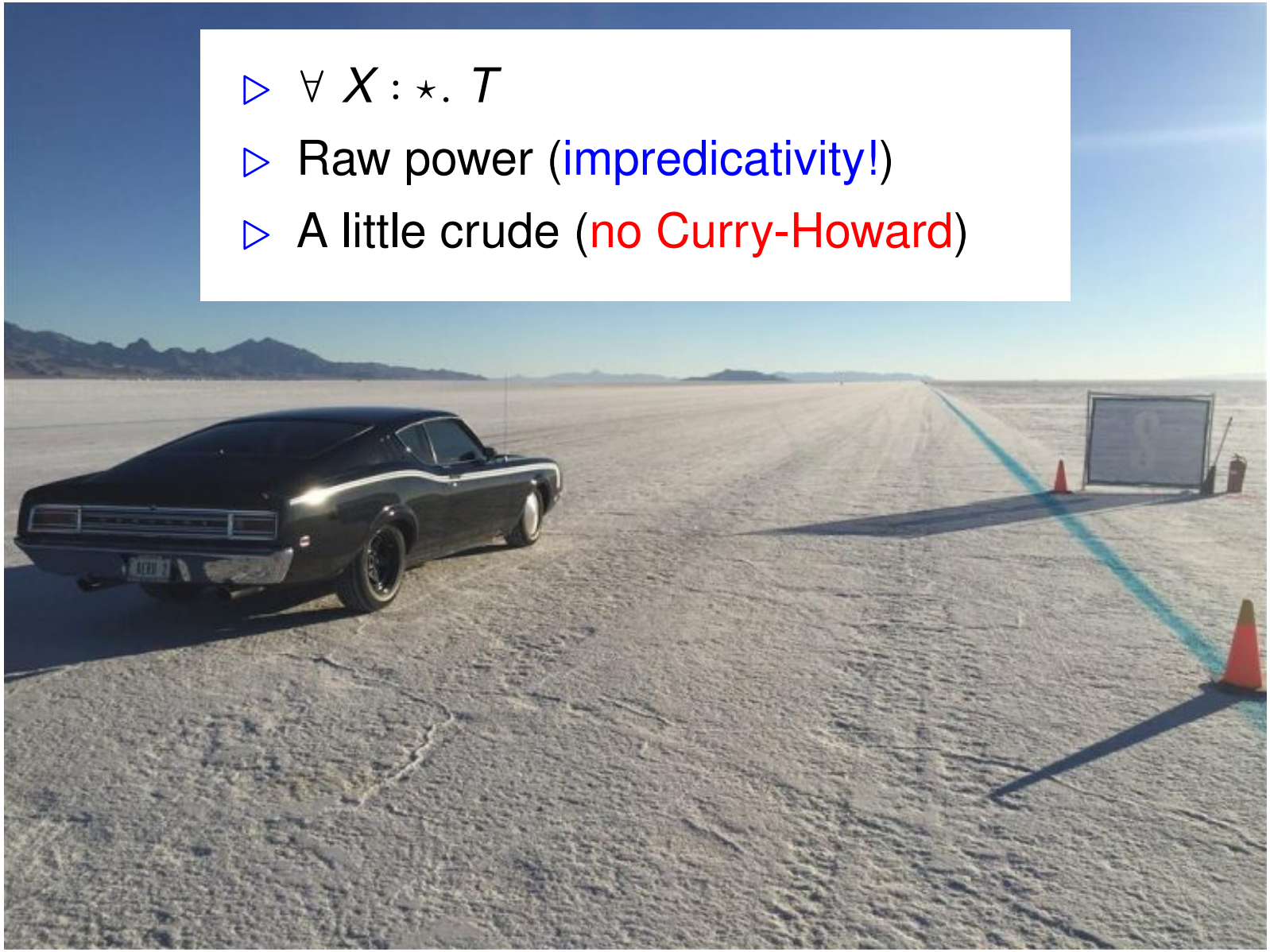
A little history

System F (Girard, Reynolds, early 1970s)

1969 Mercury Cyclone Spoiler II

System F (Girard, Reynolds, early 1970s)

- $\forall\, X : \star.\ T$
- Raw power (impredicativity!)
- A little crude (no Curry-Howard)



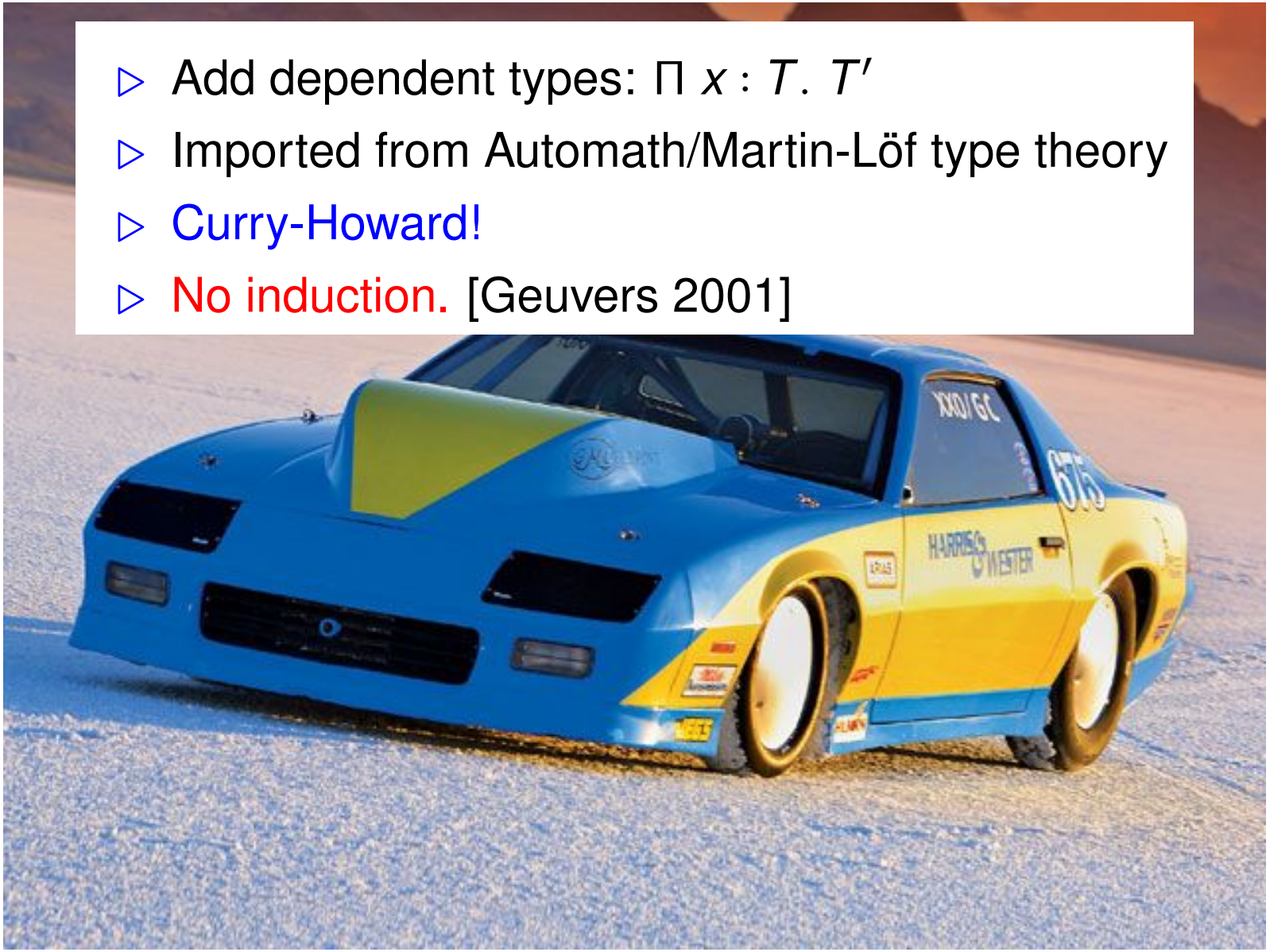
1969 Mercury Cyclone Spoiler II

Calculus of Constructions (Coquand, Huet 1988)

1988 Chevrolet Camaro

Calculus of Constructions (Coquand, Huet 1988)

- ▷ Add dependent types: $\Pi\, x : T.\ T'$
- ▷ Imported from Automath/Martin-Löf type theory
- ▷ Curry-Howard!
- ▷ No induction. [Geuvers 2001]

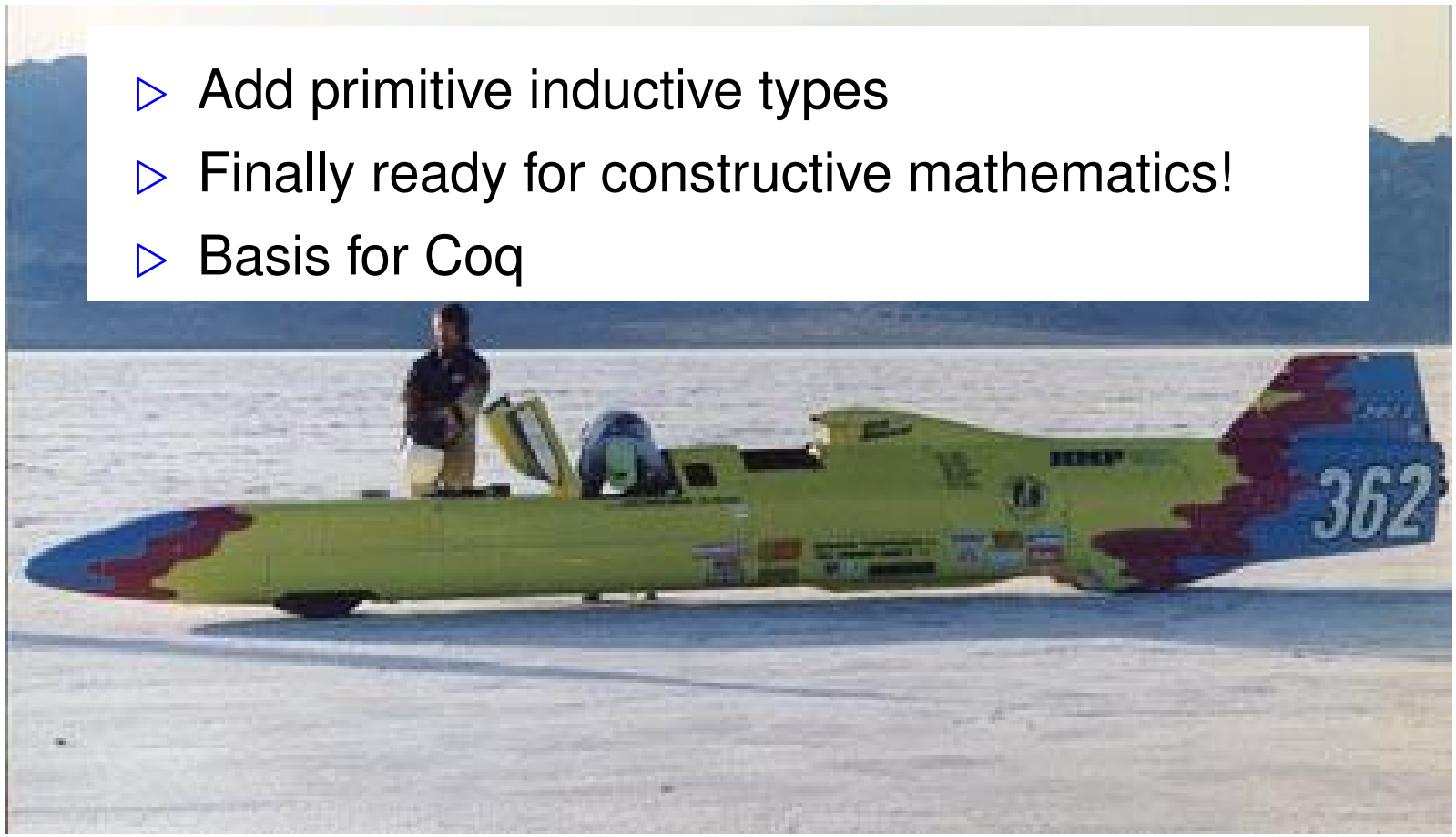


1988 Chevrolet Camaro

Calculus of Inductive Constructions (Werner 1994)

1992 Hoffman-Markley Streamliner

Calculus of Inductive Constructions (Werner 1994)



- Add primitive inductive types
- Finally ready for constructive mathematics!
- Basis for Coq

1992 Hoffman-Markley Streamliner

But Coq ≠ CIC

- Coinductive types
- Universe hierarchy (Extended CC, Luo 1990)
- Proof-irrelevant universe `Prop`
- *And we might want more:*
  - definitional proof irrelevance
  - inductive-inductive types
  - inductive-recursive types

Similarly, Agda ≠ MLTT.

Issues and limitations, Coq and Agda

- No formal semantics/correctness proof
  - Despite a lot of interest: TT in TT
- (Hence!) bugs and surprises
  - incompatibilities with various axioms
  - actual contradictions!
  - type soundness broken in Coq
- Commitment to a set of datatypes
  - theory of datatypes not finished...
  - e.g., higher-order abstract syntax prohibited

Have we created a monster?

Schaufelradbagger 258

Have we created a monster?

Schaufelradbagger 258

If I could turn back time...

Good-bye to:

- primitive datatypes
- (also universe hierarchy, my bias)

For this, hello to

- lambda-encodings of data

If I could turn back time...

Good-bye to:

- primitive datatypes
- (also universe hierarchy, my bias)

For this, hello to

- lambda-encodings of data

If I could turn back time...

Good-bye to:

- primitive datatypes
- (also universe hierarchy, my bias)

For this, hello to

- lambda-encodings of data

Wanted: a new type theory

where

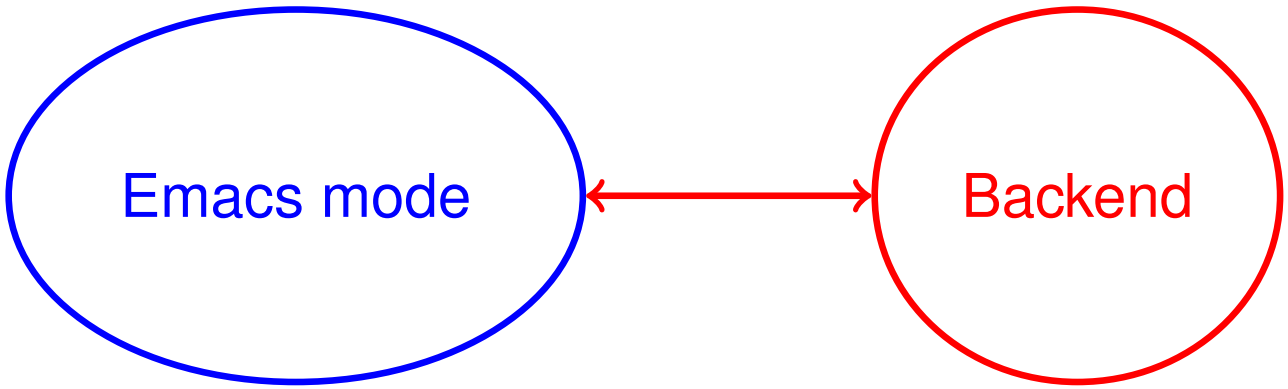
- inductive datatypes are derived (lambda-encoded)
- impredicativity is central
- core theory is small and verifiable

Tooling goals:

- see all typing/inference information
- predictable inference
- elaborate to core with independent checker

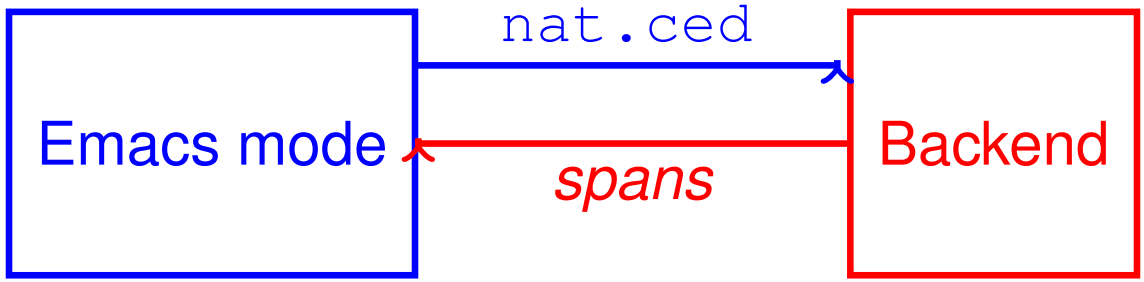
This is Cedille

Architecture of Cedille



Emacs mode (*to be revised*)

- Cedille has an emacs mode for editing Cedille files
- Based on a generic structured-editing mode by Carl Olson



- A span is [*label*, *start-pos*, *end-pos*, *attributes*]
- Spans communicated in JSON
- Cedille sends all type information, in span attributes
- Monadic style for writing the backend (type checker)