Verified Programming in Gugu

Aaron Stump[1] Morgan Deters[2] Adam Petcher[3]
Todd Schiller[3] Timothy Simpson[3]

[1]Computational Logic Center
CS, The University of Iowa

[2]LSI, Universitat Politècnica de Catalunya, Spain

[3]CSE, Washington University in St. Louis

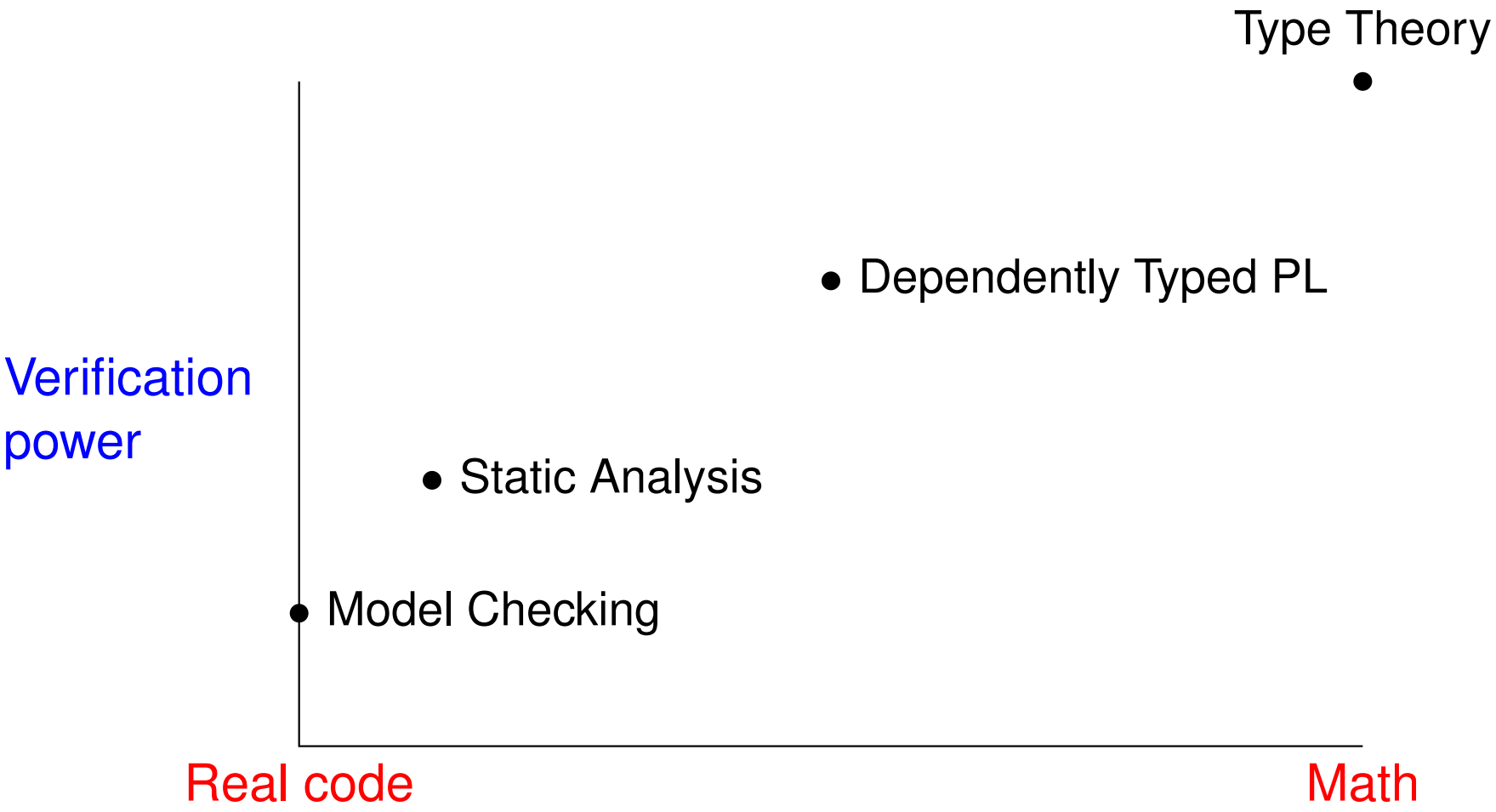
Funding from NSF CAREER.

A Vexing Continuum

Real code	Math. functions
concurrent	sequential
imperative	pure
general recursive	terminating

Where is your verification method?

Plotting Some Approaches



The GURU Approach

Real code ⇜ GURU Math. functions

General recursion
Dependently typed programs
External theorems about programs
Mutable state
No concurrency
No aliasing (yet)

Basic GURU Design

- Terms : Types.
- Proofs : Formulas.
- “Full-spectrum” dependency.
- Proofs and types can appear in terms.
  - computationally irrelevant.
  - erased by compilation, definitional equality.
- Definitional equality is very weak (no β).
- Type checking decidable.
- Explicit casts.

- Today:
  - specificational data.
  - ownership annotations.
  - functional modeling.

Specificational Data

- Programmer can designate argument positions `spec`.
  - for constructors, functions.
  - `spec x` can only be used in a `spec` position.
  - enforced separately from type checking.
- `spec` args erased by compilation [Brady+03], def. equality.
- Improves efficiency, simplifies proofs.

Example: Vector Append

```
Inductive vec : Fun(A:type)(n:nat).type :=
  vecn : Fun(A:type).<vec A Z>
| vecc : Fun(A:type)(spec n:nat)(a:A)(l:<vec A n>).
              <vec A (S n)>.

vec_append : Fun(A:type)(spec n m:nat)
                (l1 : <vec A n>)(l2 : <vec A m>).
                <vec A (plus n m)>

vec_append_assoc :
  Forall(A:type)(n1 : nat)(l1 : <vec A n1>)
        (n2 n3 : nat)(l2 : <vec A n2>)(l3 : <vec A n3>).
  { (vec_append (vec_append l1 l2) l3) =
    (vec_append l1 (vec_append l2 l3)) }
```

Compiled to C: `gvec gvec_append(gtype gA, gvec gl1, gvec gl2);`