A Proof Repository for Formal Verification of Software

Michael Franssen

WASDeTT- 3

September 20th 2010

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Cocktail

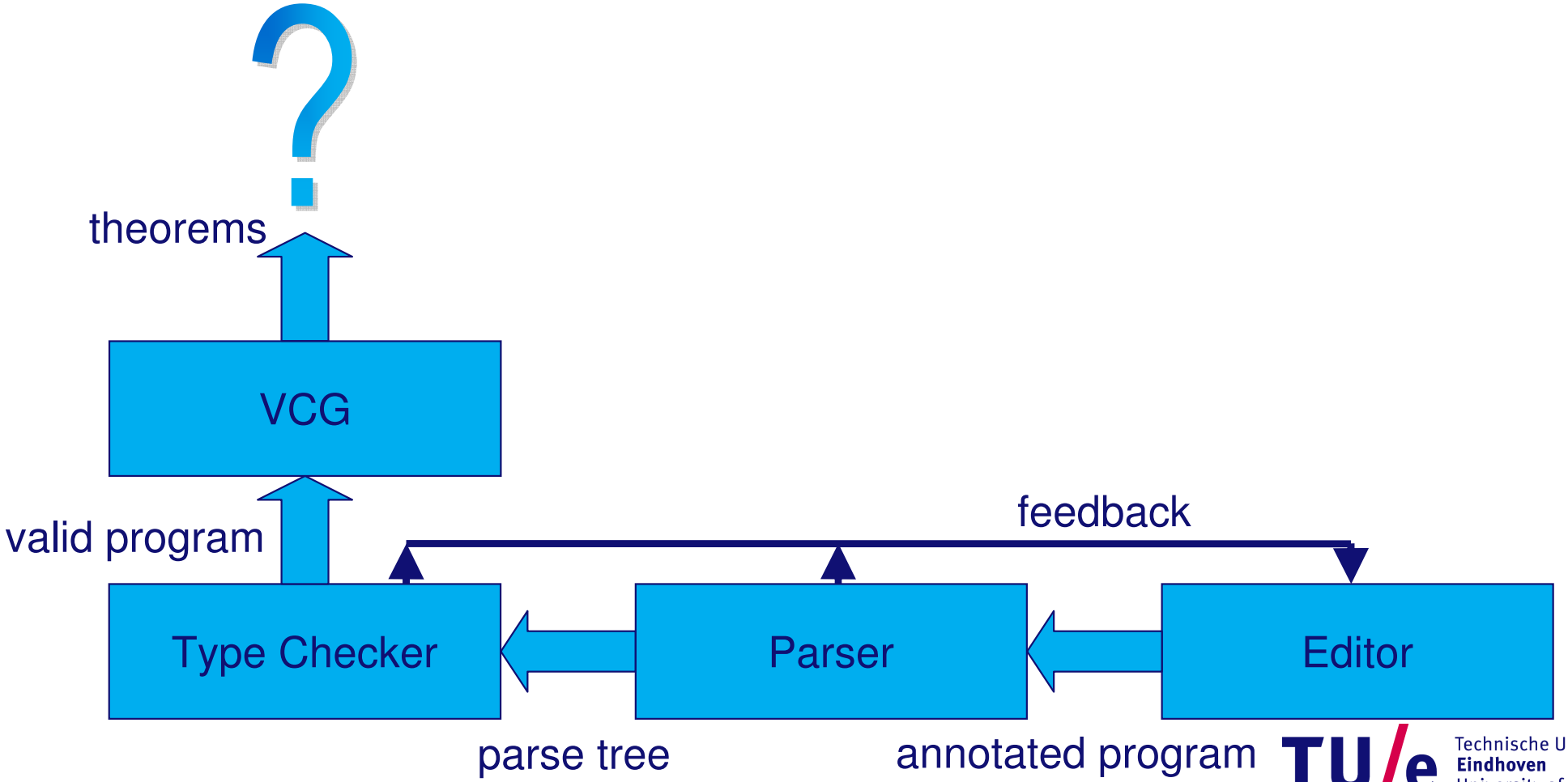
- Derive programs from their specifications

  Does not scale and nobody programs like this
- Create proofs interactively with an proof assistant based on type lambda calculus
  (with a GUI providing Fitch-style notation)

  User friendly, but much automation desired
- Custom built tableaux based automated theorem prover
  Much too weak and an awfull lot of work

Target

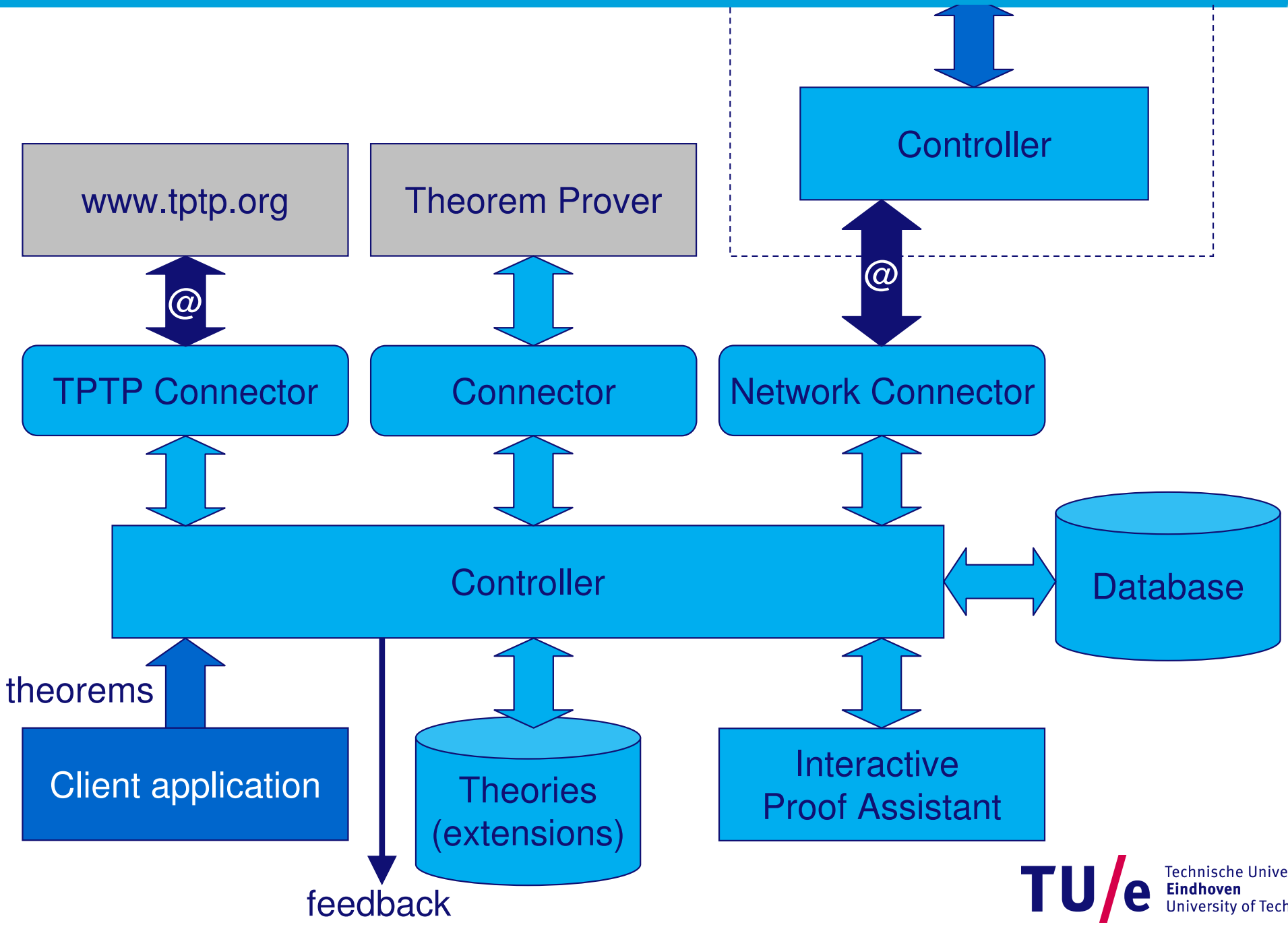
- An oracle to provide proofs required for the formal verification of software
- We assume the following architecture:



Considerations

- Do not build your own prover, but use existing ones
- Instead of choosing one prover, create a generic interface
  - Pitfall: using the greatest common divisor! (does not exploit specialized provers!)
- Automated provers are usually incomplete. What if a proof fails?
- Proving a theorem may take a while. How do we prevent proving the same theorem several times?

Our implemented modules



Techniques used

- The architecture can easiliy be configured by the user, due to our modular approach.
- If an external prover does not support some extension (e.g. integers), the required definitions and axioms are provided by the repository.
- The database uses a computable criterion called "more general" that implies "stronger". This is more flexible than looking for exact matches.
- The interactive prover has a GUI that employs proof by pointing and a Fitch style notation to enable users to conveniently construct proofs. This is done by using a typed lambda-calculus as foundation.

Questions?