Sharing Formal Mathematics and Programming

Nuprl-Light

Jason Hickey

Cornell University

Department of

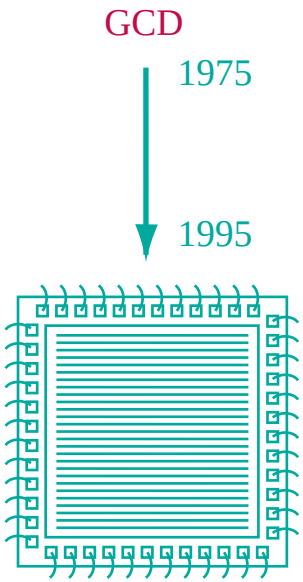
Computer Science

Sharing Formal Mathematics and Programming

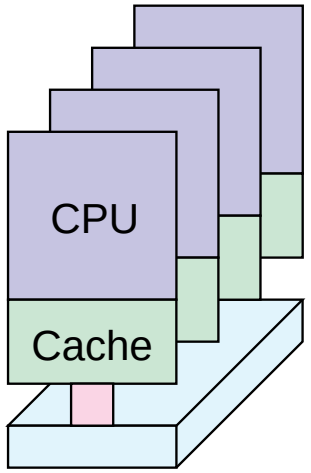
Nuprl-Light

Jason Hickey

Explosion in formal math



- **Applications**
  - AMP9000 processor in PVS
  - Cache coherency Nuprl/HOL
  - Ensemble verification
- **Mathematical Libraries**
  - QED project
  - Mizar/Journal of Formal Reasoning
  - Algebra, number theory, analysis, calculus
  - Programming semantics
  - Computing theory, automata theory



Problems & Goals

- **Problems**
  - Knowledge is formalized in different logics
  - Reasoning/tactics are logic specific
  - No mechanism for sharing work
- **Goals**
  - Meta-system for specifying and **relating** logics
  - **Mechanize** translations between theories
    - Constructive & classical
    - SoS & domain theoretic semantics
  - **Import** results of other logics
    - HOL into Nuprl

Prover architecture has not kept pace

- **Flat name spaces**
  - Every rule, theorem, axiom has a unique name
- **No abstraction/protection**
- **Single type theory**

```
RULE rule1
RULE rule2
RULE rule3
RULE rule4
RULE rule5

THM thm1_thm
THM thm2_thm
THM thm3_thm
THM thm4_thm
THM thm5_thm
THM thm6_thm
THM thm7_thm
THM thm8_thm
THM thm9_thm
THM thm10_thm
THM thm11_thm
THM thm12_thm
THM thm13_thm
THM thm14_thm
THM thm15_thm
```

Results

- **Logical foundation for formal modules**
  - Dependent records; very dependent types
- **Design of Nuprl-Light library component**
  - object-oriented capabilities based on very-dependent types
  - protection mechanism
  - abstraction
  - initial integration of tactic and logical state
- **Implementation of Nuprl-light**
  - multiple type theories
  - Ensemble integration

Scalability

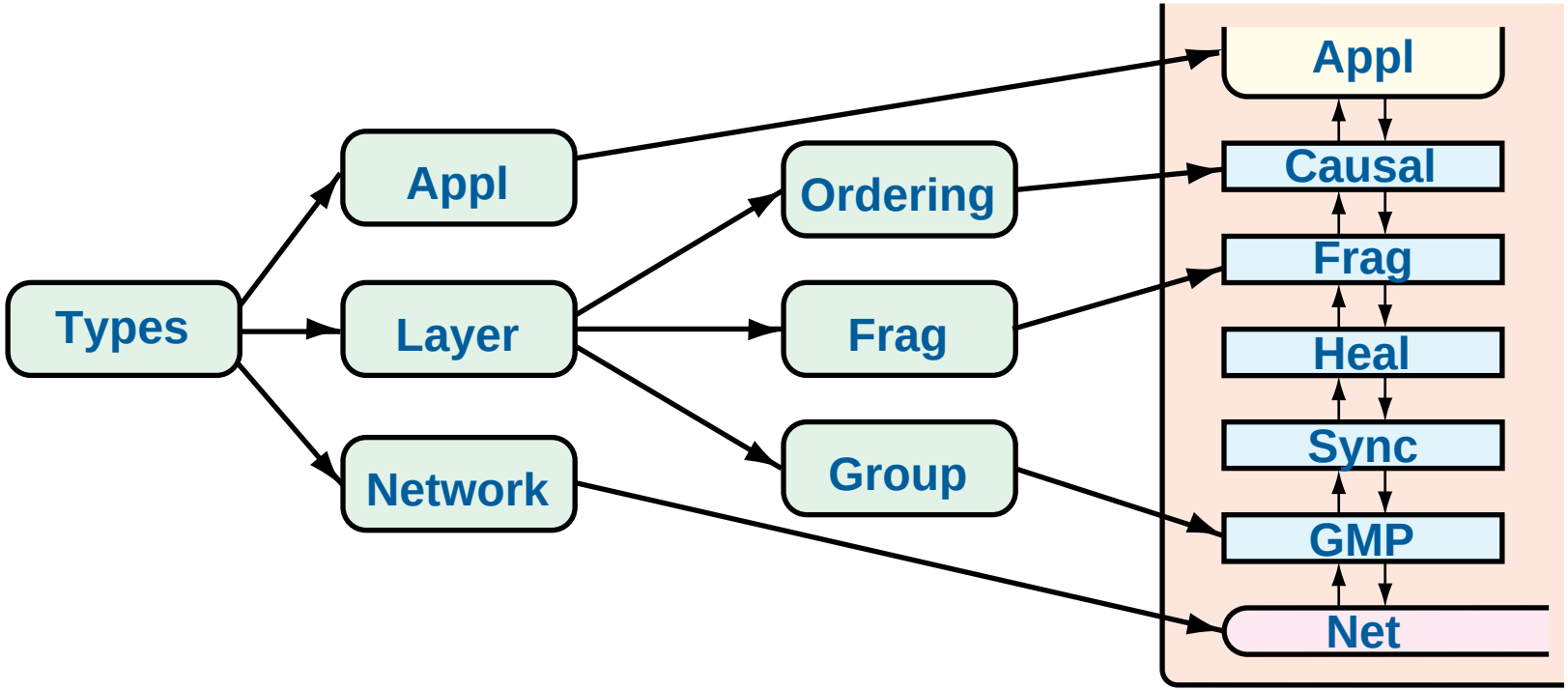
Module systems

- **Java modules provide abstraction and security**
- **ML provides second order modules**
- **Functors/sharing constraints for object oriented programming**

Hierarchy
Abstraction
Security
Naming

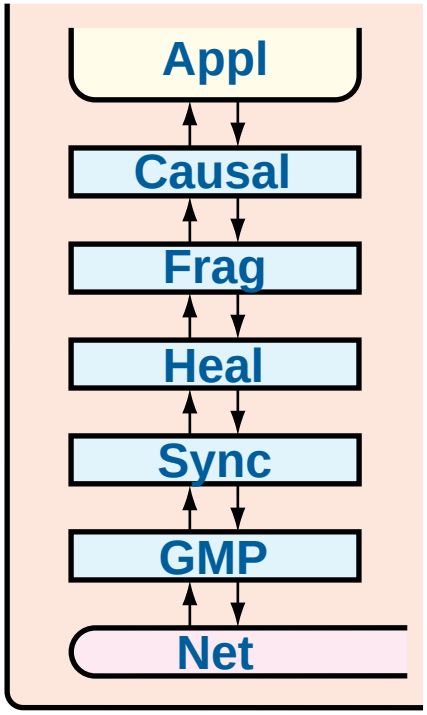
Software models

Hierarchical, refinement ordering

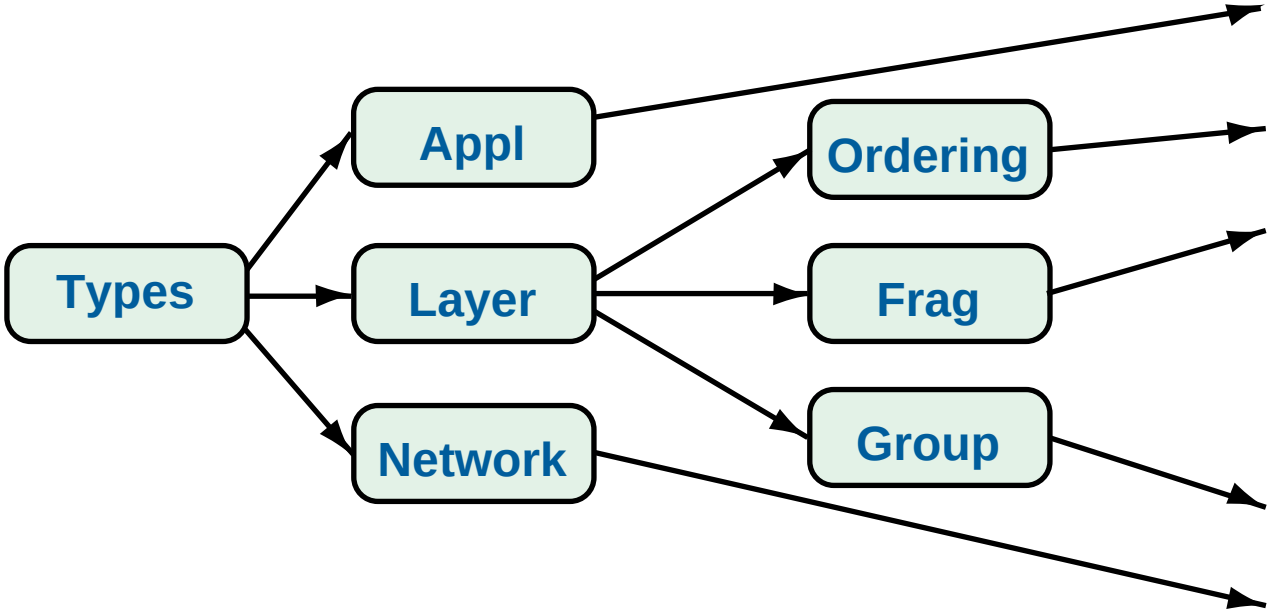


Software models

- **Hierarchical, refinement ordering**

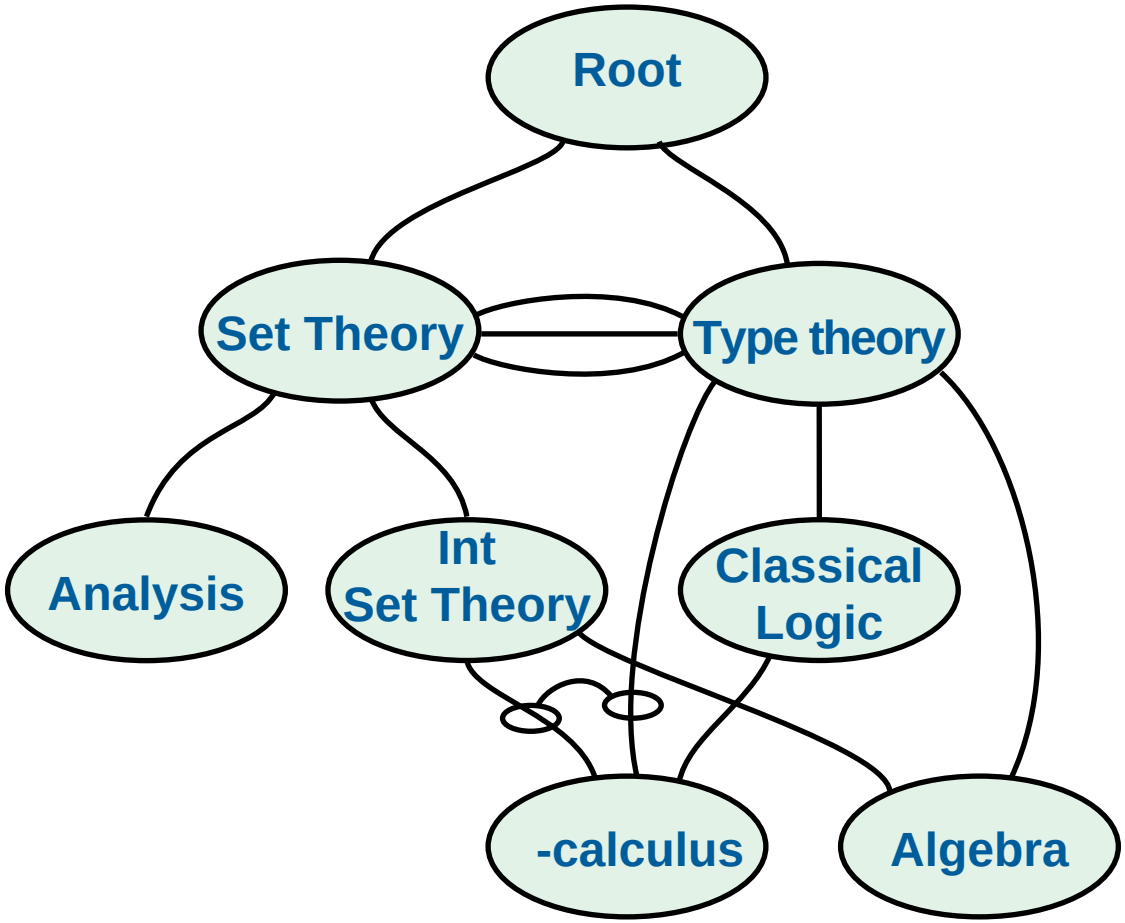


Run Time Architecture


Types
Appl
Layer
Network
Ordering
Frag
Group

Mathematical models

Arbitrary graph of entities and relationships



A Programming Logic

module type *A_LogicSig* **=**
begin

axiom inv_cond: $\dfrac{\Gamma \vdash \{I \quad P\}\ p_1\ \{I\} \qquad \Gamma \vdash \{I \qquad P\}\ p_1\ \{I\}}{\Gamma \vdash \{I\}\ (\textbf{if}\ P\ \textbf{then}\ p_1\ \textbf{else}\ p_2)\ \{I\}}$

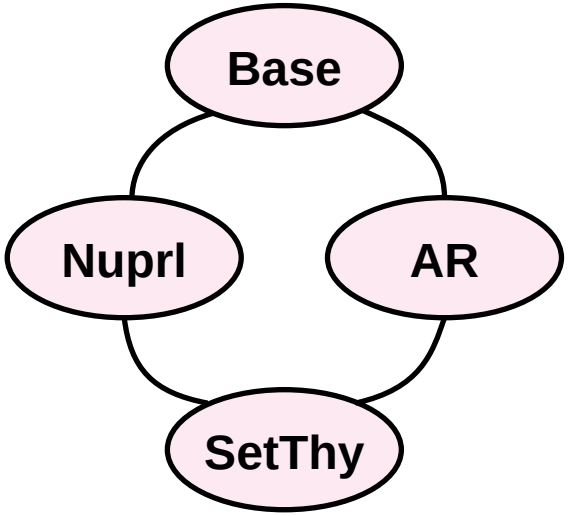
rewrite while: (**while** P **do** p **done**) (**if** P **then** p**; while** P **do** p **done**)

end

Derivations

- **Derive Intuitionistic Set theory from type theory**
- **Multiple inheritance**

```
module type ReductionSig =
begin
    include NuprlTypeTheorySig
    include AxiomOfReducibilitySig
    axiom IZFSetTheory
end
```



Status

Nuprl-Light alpha release

- **modular Nuprl type theory**
- **LF type theory**
- **automatic generation of formal module types**
- **modular tactic library**
- **Ensemble integration**

Conclusion

New model of theorem proving

- shared mathematics
- formal programming
- assistance for knowledge exchange
- theorem proving ***in-the-large***