THE PEOPLE'S REFINEMENT LOGIC

RedPRL

nuprl family & friends

nuprl family & friends

1979 Edinburgh LCF (Wadsworth, Gordon, Milner)

nuprl family & friends

1979 Edinburgh LCF (Wadsworth, Gordon, Milner)

1981 λ-PRL (Bates, Constable)

nuprl family & friends

1979 Edinburgh LCF (Wadsworth, Gordon, Milner)

1981 λ-PRL (Bates, Constable)

1982 μ-PRL (Bates, Constable)

nuprl family & friends

1979 Edinburgh LCF (Wadsworth, Gordon, Milner)

1981 λ-PRL (Bates, Constable)

1982 μ-PRL (Bates, Constable)

1985 ν-PRL/NuPRL/Nuprl (Constable, Allen, Bromley, Cleaveland, Cremer, Harper, Howe, Knoblock, Mendler, Panangaden, Sasaki, Smith)

nuprl family & friends

1979 Edinburgh LCF (Wadsworth, Gordon, Milner)

1981 λ-PRL (Bates, Constable)

1982 μ-PRL (Bates, Constable)

1985 ν-PRL/NuPRL/Nuprl (Constable, Allen, Bromley, Cleaveland, Cremer, Harper, Howe, Knoblock, Mendler, Panangaden, Sasaki, Smith)

2001 MetaPRL (Hickey, Nogin, Kopylov)

nuprl family & friends

1979 Edinburgh LCF (Wadsworth, Gordon, Milner)

1981 λ-PRL (Bates, Constable)

1982 μ-PRL (Bates, Constable)

1985 ν-PRL/NuPRL/Nuprl (Constable, Allen, Bromley, Cleaveland, Cremer, Harper, Howe, Knoblock, Mendler, Panangaden, Sasaki, Smith)

2001 MetaPRL (Hickey, Nogin, Kopylov)

2015 JonPRL (Sterling, Gratzer, Christiansen)

nuprl family & friends

1979 Edinburgh LCF (Wadsworth, Gordon, Milner)

1981 λ-PRL (Bates, Constable)

1982 μ-PRL (Bates, Constable)

1985 ν-PRL/NuPRL/Nuprl (Constable, Allen, Bromley, Cleaveland, Cremer, Harper, Howe, Knoblock, Mendler, Panangaden, Sasaki, Smith)

2001 MetaPRL (Hickey, Nogin, Kopylov)

2015 JonPRL (Sterling, Gratzer, Christiansen)

2016 **RedPRL** (Sterling, Morrison, Gratzer, Akentyev, Tosun, you?)

nuprl family & friends

1979 Edinburgh LCF (Wadsworth, Gordon, Milner)

1981 λ-PRL (Bates, Constable)

1982 μ-PRL (Bates, Constable)

1985 ν-PRL/NuPRL/Nuprl (Constable, Allen, Bromley, Cleaveland, Cremer, Harper, Howe, Knoblock, Mendler, Panangaden, Sasaki, Smith)

2001 MetaPRL (Hickey, Nogin, Kopylov)

2015 JonPRL (Sterling, Gratzer, Christiansen)

2016 **RedPRL** (Sterling, Morrison, Gratzer, Akentyev, Tosun, you?)

2016 Pudding (Christiansen)

⋮

nuprl family & friends

- 1979 Edinburgh LCF (Wadsworth, Gordon, Milner)
- 1981 λ–PRL (Bates, Constable)
- 1982 μ–PRL (Bates, Constable)
- 1985 ν–PRL/NuPRL/Nuprl (Constable, Allen, Bromley, Cleaveland, Cremer, Harper, Howe, Knoblock, Mendler, Panangaden, Sasaki, Smith)
- 2001 MetaPRL (Hickey, Nogin, Kopylov)
- 2015 JonPRL (Sterling, Gratzer, Christiansen)
- 2016 **RedPRL** (Sterling, Morrison, Gratzer, Akentyev, Tosun, you?)
- 2016 Pudding (Christiansen)

⋮

THE PRL FAMILY IS THE VANGUARD OF THE PEOPLE!

nominal abstract binding trees

- ⋆ syntax with binding (second order universal algebra)

nominal abstract binding trees

- ⋆ syntax with binding (second order universal algebra)
- ⋆ both variables and symbols (nominal atoms)

nominal abstract binding trees

- ⋆ syntax with binding (second order universal algebra)
- ⋆ both variables and symbols (nominal atoms)
- ⋆ nominal sets, sheaf semantics

nominal abstract binding trees

- ⋆ syntax with binding (second order universal algebra)
- ⋆ both variables and symbols (nominal atoms)
- ⋆ nominal sets, sheaf semantics
- ⋆ crucial for: local exceptions, store, cubical dimensions

nominal abstract binding trees

- ⋆ syntax with binding (second order universal algebra)
- ⋆ both variables and symbols (nominal atoms)
- ⋆ nominal sets, sheaf semantics
- ⋆ crucial for: local exceptions, store, cubical dimensions
- ⋆ introduced by Harper in Practical Foundations for Programming Languages

computational type theory

Set-theoretic semantics:

$$\mathcal{J} \in \mathbf{SET} ::= \left|\begin{array}{l} A = B\ \mathit{type} \\ M = N \in A \\ M \preccurlyeq N \\ \ldots \end{array}\right. \qquad \text{(categorical judgment)}$$

computational type theory

Set-theoretic semantics:

$$\mathcal{J} \in \mathbf{SET} ::= \left| \begin{array}{l} A = B \ \mathit{type} \\ M = N \in A \\ M \preccurlyeq N \\ \dots \end{array} \right.$$

(categorical judgment)

$$\mathcal{J}_2 \ (\mathcal{J}_1) \triangleq {\mathcal{J}_2}^{\mathcal{J}_1}$$

(hypothetical judgment)

$$|_{\mathfrak{x}:\tau} \ \mathcal{J}[\mathfrak{x}] \triangleq \prod_{m \in \tau} \mathcal{J}[m]$$

(general judgment)

computational type theory

Set-theoretic semantics:

$$\mathcal{J} \in \mathsf{SET} ::= \left| \begin{array}{l} A = B\ \mathit{type} \\ M = N \in A \\ M \preccurlyeq N \\ \ldots \end{array} \right.$$

(categorical judgment)

$$\mathcal{J}_2\ (\mathcal{J}_1) \triangleq \mathcal{J}_2^{\mathcal{J}_1}$$

(hypothetical judgment)

$$|_{\mathfrak{X}:\tau}\ \mathcal{J}[\mathfrak{X}] \triangleq \prod_{m \in \tau} \mathcal{J}[m]$$

(general judgment)

$$\Gamma \vDash M = N \in A \triangleq |_{\gamma_0,\gamma_1:|\Gamma|}\ M\gamma_0 = N\gamma_1 \in A\gamma_0\ \ (\gamma_0 = \gamma_1 \in \Gamma)$$

(sequent judgment)

nominal computational type theory

Sheaf-theoretic semantics, let $\mathbb{I}[\mathcal{S}] \triangleq \left((\mathbb{I} \downarrow \mathcal{S})^{\mathrm{o}}, J_{\mathtt{atomic}}\right)$:

$$\mathcal{J} \in \mathbf{Sh}(\mathbb{I}[\mathcal{S}]) ::= \left| \begin{array}{l} A = B \ \mathit{type} \\ M = N \in A \\ M \preccurlyeq N \\ \cdots \end{array} \right.$$ (parametric judgment)

$$\mathcal{J}_2 \ (\mathcal{J}_1) \triangleq {\mathcal{J}_2}^{\mathcal{J}_1}$$ (hypothetical judgment)

$$|_{\text{Ӿ}:\tau} \ \mathcal{J}[\text{Ӿ}] \triangleq \prod_{m \in \tau} \mathcal{J}[m]$$ (general judgment)

$$\Gamma \vDash M = N \in A \triangleq |_{\gamma_0, \gamma_1 : |\Gamma|} \ M\gamma_0 = N\gamma_1 \in A\gamma_0 \ \ (\gamma_0 = \gamma_1 \in \Gamma)$$ (sequent judgment)

nominal computational type theory

Sheaf-theoretic semantics, let $\mathbb{I}[\mathcal{S}] \triangleq \left((\mathbb{I} \downarrow \mathcal{S})^{\mathsf{o}}, J_{\mathsf{atomic}}\right)$:

$$\mathcal{J} \in \mathbf{Sh}(\mathbb{I}[\mathcal{S}]) ::= \left| \begin{array}{l} A = B\ \mathit{type} \\ M = N \in A \\ M \preccurlyeq N \\ \ldots \end{array} \right.$$

(*parametric* judgment)

$$\Upsilon \Vdash \mathcal{J}_2 \ \ (\mathcal{J}_1) \triangleq [\mathbf{y}^{\mathsf{o}}(\Upsilon) \times \mathcal{J}_1, \mathcal{J}_2]$$

(hypothetical judgment)

$$\Upsilon \Vdash |_{\text{ъ}:\tau} \ \mathcal{J}[\text{ъ}] \triangleq \left\{ s \in \left[\mathbf{y}^{\mathsf{o}}(\Upsilon) \times \tau, \oint \mathcal{J}\right] \mid \pi_{\mathcal{J}} \circ s = \mathsf{id} \right\}$$

(general judgment)

$$\Gamma \vDash M = N \in A \triangleq |_{\gamma_0,\gamma_1:|\Gamma|} \ M\gamma_0 = N\gamma_1 \in A\gamma_0 \ \ (\gamma_0 = \gamma_1 \in \Gamma)$$

(sequent judgment)

nominal computational type theory

- ⋆ reason about untyped computational approximation, congruence(!)

nominal computational type theory

- ⋆ reason about untyped computational approximation, congruence(!)
- ⋆ strict/exact equality (functions, quotients, PERs, truncation)

nominal computational type theory

- ⋆ reason about untyped computational approximation, congruence(!)
- ⋆ strict/exact equality (functions, quotients, PERs, truncation)
- ⋆ compositional dependent records (better than Agda, Coq!)

nominal computational type theory

- ⋆ reason about untyped computational approximation, congruence(!)
- ⋆ strict/exact equality (functions, quotients, PERs, truncation)
- ⋆ compositional dependent records (better than Agda, Coq!)
- ⋆ partiality, general recursion (universal computation)

nominal computational type theory

- ⋆ reason about untyped computational approximation, congruence(!)
- ⋆ strict/exact equality (functions, quotients, PERs, truncation)
- ⋆ compositional dependent records (better than Agda, Coq!)
- ⋆ partiality, general recursion (universal computation)
- ⋆ fresh name generation as a computational effect!

nominal computational type theory

- ⋆ reason about untyped computational approximation, congruence(!)
- ⋆ strict/exact equality (functions, quotients, PERs, truncation)
- ⋆ compositional dependent records (better than Agda, Coq!)
- ⋆ partiality, general recursion (universal computation)
- ⋆ fresh name generation as a computational effect!
- ⋆ **orthodox** (*Brouwerian*) intuitionistic continuity principles, Bar Theorem, Fan Theorem, free choice sequences

nominal computational type theory

- ⋆ reason about untyped computational approximation, congruence(!)
- ⋆ strict/exact equality (functions, quotients, PERs, truncation)
- ⋆ compositional dependent records (better than Agda, Coq!)
- ⋆ partiality, general recursion (universal computation)
- ⋆ fresh name generation as a computational effect!
- ⋆ **orthodox** (*Brouwerian*) intuitionistic continuity principles, Bar Theorem, Fan Theorem, free choice sequences

DECISIVELY SMASH THE FORMALIST CLIQUE!


PRL
WE CAN PROVE IT

dependent refinement

dependent refinement

Nupri, MetaPRL, JonPRL:

$$\frac{H, x : A \gg (B[x] \in \mathbb{U}_i)\ \mathit{true} \rightsquigarrow \lfloor \star \rfloor \qquad \begin{array}{l} H \gg (M \in A)\ \mathit{true} \rightsquigarrow \lfloor \star \rfloor \\ H \gg B[M]\ \mathit{true} \rightsquigarrow \lfloor N \rfloor \end{array}}{H \gg \sum_{(x \in A)} B[x]\ \mathit{true} \rightsquigarrow \lfloor \langle M, N \rangle \rfloor}\ \textstyle\sum\text{-intro}\{M, i\}$$

No general refinement rules expressible for $\sum$-introduction, $\prod$-elimination ("constructible subgoals property")

dependent refinement

RedPRL:

$$\frac{H, x : A \gg B[x]\ \mathit{type} \rightsquigarrow \lfloor i \rfloor \qquad \begin{array}{l} H \gg A\ \mathit{true} \rightsquigarrow \lfloor M \rfloor \\ H \gg B[M]\ \mathit{true} \rightsquigarrow \lfloor N \rfloor \end{array}}{H \gg \sum_{(x \in A)} B[x]\ \mathit{true} \rightsquigarrow \lfloor \langle M, N \rangle \rfloor}\ \sum\text{-intro}$$

Dependent LCF allows subgoals to depend on the syntheses of previous subgoals, enabling <u>bidirectional refinement</u>.

dependent refinement

RedPRL:

$$\frac{H, x : A \gg B[x] \; \mathit{type} \rightsquigarrow \lfloor i \rfloor \qquad \begin{array}{c} H \gg A \; \mathit{true} \rightsquigarrow \lfloor M \rfloor \\ H \gg B[M] \; \mathit{true} \rightsquigarrow \lfloor N \rfloor \end{array}}{H \gg \sum_{(x \in A)} B[x] \; \mathit{true} \rightsquigarrow \lfloor \langle M, N \rangle \rfloor} \; \textstyle\sum\text{-intro}$$

Dependent LCF allows subgoals to depend on the syntheses of previous subgoals, enabling <u>bidirectional refinement</u>.

CRITICIZE THE OLD WORLD AND BUILD A NEW WORLD WITH JON STERLING THOUGHT AS A WEAPON!

practical implementation

practical implementation

- ⋆ $\approx$ 9000 lines of Standard ML (MLton, SML/NJ)

practical implementation

- ⋆ $\approx$ 9000 lines of **Standard ML** (MLton, SML/NJ)
- ⋆ highly modular, library-oriented design & specification

practical implementation

- ⋆ $\approx$ 9000 lines of **Standard ML** (MLton, SML/NJ)
- ⋆ highly modular, library-oriented design & specification
- ⋆ standardized implementation language $\Longrightarrow$ builds for perpetuity

practical implementation

- ⋆ $\approx$ 9000 lines of Standard ML (MLton, SML/NJ)
- ⋆ highly modular, library-oriented design & specification
- ⋆ standardized implementation language $\Longrightarrow$ builds for perpetuity
- ⋆ much left to do: add more rules, experiment with cubical realizability, improve frontend,...

practical implementation

- ⋆ $\approx$ 9000 lines of Standard ML (MLton, SML/NJ)
- ⋆ highly modular, library-oriented design & specification
- ⋆ standardized implementation language $\Longrightarrow$ builds for perpetuity
- ⋆ much left to do: add more rules, experiment with cubical realizability, improve frontend,...
- ⋆ easy to install, no database required!

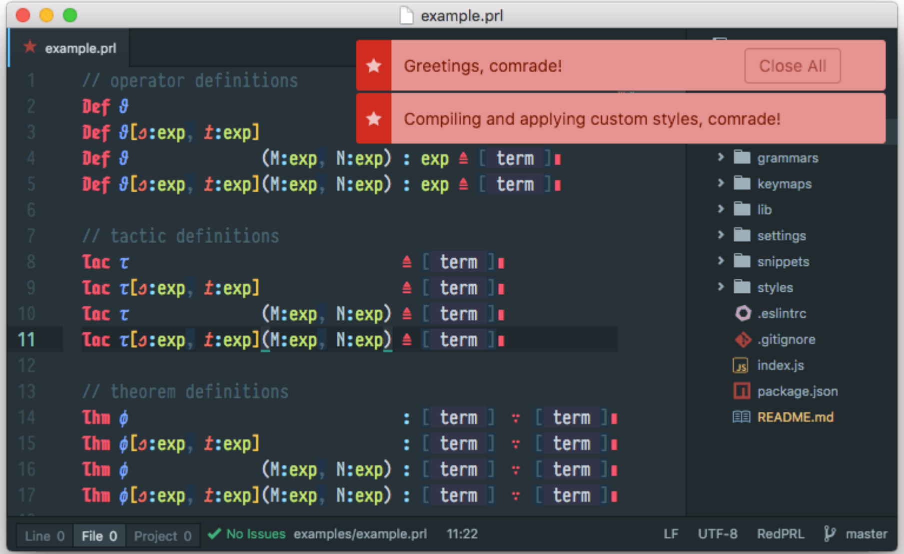
practical implementation

- ⋆ $\approx$ 9000 lines of Standard ML (MLton, SML/NJ)
- ⋆ highly modular, library-oriented design & specification
- ⋆ standardized implementation language $\Longrightarrow$ builds for perpetuity
- ⋆ much left to do: add more rules, experiment with cubical realizability, improve frontend,...
- ⋆ easy to install, no database required!
- ⋆ See redprl.org & github.com/redprl; join the fun!

practical implementation

- ≈ 9000 lines of Standard ML (MLton, SML/NJ)
- highly modular, library-oriented design & specification
- standardized implementation language $\Longrightarrow$ builds for perpetuity
- much left to do: add more rules, experiment with cubical realizability, improve frontend,...
- easy to install, no database required!
- See redprl.org & github.com/redprl; join the fun!

OPEN-SOURCE CONTRIBUTORS ARE THE SHOCK TROOPS OF THE REVOLUTION!


example.prl
example.prl
Greetings, comrade!
Close All
Compiling and applying custom styles, comrade!
// operator definitions
// tactic definitions
// theorem definitions
grammars
keymaps
lib
settings
snippets
styles
.eslintrc
.gitignore
index.js
package.json
README.md
Line 0
File 0
Project 0
No Issues
examples/example.prl
11:22
LF
UTF-8
RedPRL
master


COQ