Recursive Rows in Rome

AH & JGM

ACM Reference Format:

AH & JGM. 2023. Recursive Rows in Rome. 1, 1 (December 2023), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 IX: THE INDEX CALCULUS

1.1 Syntax

Let $0, 1, 2, \ldots$ denote object-level natural numbers in the intuitive fashion and let i_n be the finite natural obtained by n applications of FSuc to FZero.

$$
\begin{array}{llcl}
\text{Sorts} & \sigma & ::= & \star \mid \mathcal{U} \\
\text{Terms} & A, B, M, N, T & ::= & \star \mid x \mid \\
 & & & \text{Nat} \mid \text{Zero} \mid \text{Suc}\, M \mid \\
 & & & \text{case}_{\mathbb{N}}\, M\, \text{of}\, \{\text{Zero} \mapsto N_1; \text{Suc}\, x \mapsto N_2\} \mid \\
 & & & \text{Ix}\, M \mid \text{FZero} \mid \text{FSuc}\, M \mid \\
 & & & \text{case}_{\text{Fin}}\, M\, \text{of}\, \{\text{FZero} \mapsto N_1; \text{FSuc}\, x \mapsto N_2\} \mid \\
 & & & \{\!\{M_1, \ldots, M_n\}\!\} \mid \\
 & & & \top \mid \mathtt{tt} \mid \\
 & & & \forall \alpha : T.N \mid \lambda x : T.N \mid M\, N \mid \\
 & & & \exists \alpha : T.M \mid (\alpha : T, M) \mid \\
 & & & M + N \mid \text{left}\, M \mid \text{right}\, M \mid \\
 & & & \text{case}_{+}\, M\, \text{of}\, \{\text{left}\, x \mapsto N_1; \text{right}\, y \mapsto N_2\} \mid \\
 & & & M \equiv N \mid \text{refl}\, T\, M\, N \mid \\
\text{Environments} & \Gamma & ::= & \varepsilon \mid \Gamma, \alpha : T
\end{array}
$$

Fig. 1. Syntax

The syntax $\{\!\{M_1, \ldots, M_n\}\!\}$ is syntactic sugar for the large elimination of finite naturals on a known number of cases, defined recursively: (fix this later... multiple routes. Technically the absurd pattern has in its context an absurd assumption (that 1 == 2).) But maybe can just fix this with typing rules.

Actually it will be better to just introduce a singleton elimination form.

Author's address: AH & JGM.

XXXX-XXXX/2023/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

$$\{\!\{M\}\!\} = \lambda x : \mathrm{Ix}\,1.\mathrm{case}_{\mathrm{Fin}}\; x \;\mathrm{of}\, \{\mathrm{FZero} \mapsto M; \mathrm{FSuc}\, \bot \mapsto \bot\}$$

$$\{\!\{M_1, ..., M_n\}\!\} =_{def} \lambda(x : \mathrm{Ix}\, n).\mathrm{case}_{\mathrm{Fin}}\; x\; \mathrm{of}\, \{\mathrm{FZero} \mapsto M_1; \mathrm{FSuc}\, x \mapsto \mathrm{case}_{\mathrm{Fin}}\; x\; \mathrm{of}\, \{\mathrm{FZero} \mapsto M_2; \mathrm{FSuc}\, b \mapsto u\}\, tts\}$$

1.2 Typing

$$\boxed{\vdash \Gamma}$$

$$(\textsc{emp})\,\frac{}{\vdash \varepsilon} \qquad (\textsc{var})\,\frac{\vdash \Gamma \quad \Gamma \vdash M : \sigma}{\vdash \Gamma, x : M}$$

$$\boxed{\Gamma \vdash M : \sigma}$$

$$(\star)\,\frac{}{\Gamma \vdash \star : \mathcal{U}} \qquad (\top)\,\frac{}{\Gamma \vdash \top : \sigma} \qquad (\textsc{Nat})\,\frac{}{\Gamma \vdash \mathrm{Nat} : \star} \qquad (\mathrm{Ix})\,\frac{\Gamma \vdash n : \mathrm{Nat}}{\Gamma \vdash \mathrm{Ix}\, n : \star}$$

$$(\forall)\,\frac{\Gamma \vdash M : \sigma_1 \quad \Gamma, \alpha : M \vdash N : \sigma_2}{\Gamma \vdash \forall \alpha : M.N : \sigma_2} \qquad (\exists)\,\frac{\Gamma \vdash M : \sigma_1 \quad \Gamma, \alpha : M \vdash N : \sigma_2}{\Gamma \vdash \exists \alpha : M.N : \sigma_2}$$

$$(+)\,\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M + N : \sigma} \qquad (\equiv)\,\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M \equiv N : \sigma}$$

Fig. 2. Context and type formation rules

$$\boxed{\Gamma \vdash M : N}$$

$$(\textsc{Var})\ \frac{x : M \in \Gamma}{\Gamma \vdash x : M} \qquad (\texttt{tt})\ \frac{}{\Gamma \vdash \texttt{tt} : \top}$$

$$(\text{Zero})\ \frac{}{\Gamma \vdash \text{Zero} : \text{Nat}} \qquad (\text{Suc})\ \frac{\Gamma \vdash n : \text{Nat}}{\Gamma \vdash \text{Suc}\ n : \text{Nat}}$$

$$(\text{FZero})\ \frac{\Gamma \vdash n : \text{Nat}}{\Gamma \vdash \text{FZero} : \text{Ix}\ (\text{Suc}\ n)} \qquad (\text{FSuc})\ \frac{\Gamma \vdash n : \text{Nat} \quad \Gamma \vdash i : \text{Ix}\ n}{\Gamma \vdash \text{FSuc}\ i : \text{Ix}\ (\text{Suc}\ n)}$$

$$(\forall I)\ \frac{\Gamma \vdash T : \star \quad \Gamma, x : T \vdash M : N}{\Gamma \vdash \lambda x : T.M : \forall(x : T).N} \qquad (\forall E)\ \frac{\Gamma \vdash M : \forall(x : T_1).T_2 \quad \Gamma \vdash N : T_1}{\Gamma \vdash M\,N : T_2[N/x]}$$

$$(\exists I)\ \frac{\Gamma \vdash M : T_1 \quad \Gamma \vdash N : T_2[M/x]}{\Gamma \vdash (M : T_1, N) : \exists(x : T_1).T_2} \qquad (\exists E_1)\ \frac{\Gamma \vdash M : \Sigma(x : T_1).T_2}{\Gamma \vdash \text{fst}\ M : T_1} \qquad (\exists E_2)\ \frac{\Gamma \vdash M : \Sigma(x : T_1).T_2}{\Gamma \vdash \text{snd}\ M : T_1[\text{fst}\ M/x]}$$

$$(\equiv I)\ \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \text{refl} : M \equiv M} \qquad (\textsc{conv})\ \frac{\Gamma \vdash M : T_1 \quad \Gamma \vdash T_1 = T_2 : \sigma \quad \sigma\ \text{Safe}}{\Gamma \vdash M : T_2}$$

$$(\equiv E)\ \frac{\begin{array}{c}\Gamma{\vdash}P{:}T_1{\equiv}T_2\\ \Gamma{\vdash}M{:}T_1\\ \Gamma{\vdash}N{:}T_1\\ \Gamma,x{:}T_1,y{:}T_1,p{:}x{\equiv}y{\vdash}T{:}\star\\ \Gamma,z{:}T_1{\vdash}H{:}T[z/x,z/y,\text{refl}/p]\end{array}}{\Gamma \vdash \mathcal{J}\ H\ M\ N\ P : T[M/x, N/y, P/p]}$$

Fig. 3. Typing rules. Missing nat, fin, and sum elimination. Fin elimination should have special case for Ix 1.

$$\boxed{\Gamma \vdash M = N : \sigma}$$

$$(\textsc{e-refl})\ \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M = M : \sigma} \qquad (\textsc{e-sym})\ \frac{\Gamma \vdash N = M : \sigma}{\Gamma \vdash M = N : \sigma} \qquad (\textsc{e-trans})\ \frac{\Gamma \vdash M = P : \sigma \quad \Gamma \vdash P = N : \sigma}{\Gamma \vdash M = N : \sigma}$$

$$\boxed{\Gamma \vdash M = N : T}$$

$$(\textsc{c-refl})\ \frac{\Gamma \vdash M : T}{\Gamma \vdash M = M : T} \qquad (\textsc{c-sym})\ \frac{\Gamma \vdash N = M : T}{\Gamma \vdash M = N : T} \qquad (\textsc{c-trans})\ \frac{\Gamma \vdash M = P : T \quad \Gamma \vdash P = N : T}{\Gamma \vdash M = N : T}$$

Fig. 4. Definitional equality & computational laws

1.3 A Comparison to $\lambda^{\Pi\mathcal{U}\mathbb{N}}$ [Abel et al. 2018]

2 TRANSLATION FROM Rω

2.1 Untyped Translation

We follow the approach of [Morris and McKinna 2019] and give both typed and untyped translations of Rω types. Figure 6 describe the untyped translation, which is used to show translational soundness of the typed translation (Figure 6).

$$\boxed{(\kappa)^\bullet}$$

$$\begin{aligned}
(\star)^\bullet &= \star \\
(\mathsf{L})^\bullet &= \top \\
(\kappa_1 \to \kappa_2)^\bullet &= \forall(\alpha : (\kappa_1)^\bullet).(\kappa_2)^\bullet \\
(\mathsf{R}^\kappa)^\bullet &= \exists(n : \mathrm{Nat}).\forall(j : \mathrm{Ix}\; n).(\kappa)^\bullet
\end{aligned}$$

$$\boxed{(\Gamma \vdash \tau : \kappa)^\bullet}$$

$$\boxed{\text{(When } \kappa \text{ not row-kinded.)}}$$

$$\begin{aligned}
(\alpha)^\bullet &= \alpha \\
(\tau_1 \to \tau_2)^\bullet &= \forall(\alpha : (\tau_1)^\bullet).(\tau_2)^\bullet \\
(\forall\alpha : \kappa.\tau)^\bullet &= \forall(\alpha : (\kappa)^\bullet).(\tau)^\bullet \\
(\lambda\alpha : \kappa.\tau)^\bullet &= \forall(\alpha : (\kappa)^\bullet).(\tau)^\bullet \\
(\pi \Rightarrow \tau)^\bullet &= \forall(\alpha : (\pi)^\bullet).(\tau)^\bullet \\
(\tau\, \upsilon)^\bullet &= (\tau)^\bullet\, (\upsilon)^\bullet \\
(\ell)^\bullet &= \top \\
(\lfloor \xi \rfloor)^\bullet &= \top \\
(\xi \triangleright \tau)^\bullet &= (\tau)^\bullet \\
(\Pi\rho)^\bullet &= \forall(i : \mathrm{Ix}\,(\mathrm{fst}\,(\rho)^\bullet)).(\mathrm{snd}\,(\rho)^\bullet)\; i \\
(\Sigma\rho)^\bullet &= \exists(i : \mathrm{Ix}\,(\mathrm{fst}\,(\rho)^\bullet)).(\mathrm{snd}\,(\rho)^\bullet)\; i
\end{aligned}$$

$$\boxed{\text{(When } \kappa \text{ row-kinded.)}}$$

$$\begin{aligned}
(\rho\, \lceil \upsilon \rceil)^\bullet &= (\mathrm{fst}\,(\rho)^\bullet : \mathrm{Nat}, \lambda(j : \mathrm{Ix}\,(\mathrm{fst}\,(\rho)^\bullet)).(\mathrm{snd}\,(\rho)^\bullet)(\upsilon)^\bullet) \\
(\lceil \tau \rceil\, \rho)^\bullet &= (\mathrm{fst}\,(\rho)^\bullet : \mathrm{Nat}, \lambda(j : \mathrm{Ix}\,(\mathrm{fst}\,(\rho)^\bullet)).(\tau)^\bullet\,(\mathrm{snd}\,(\rho)^\bullet)) \\
(\xi \triangleright_{\mathsf{R}} \tau)^\bullet &= (1,\; \{\!\!\{ (\tau)^\bullet \}\!\!\})
\end{aligned}$$

$$\boxed{(\Gamma \vdash \pi : \kappa)^\bullet}$$

$$\ldots$$

Fig. 5. A compositional translation of typed Rω kinds and predicates to untyped Ix terms

$$\boxed{(\Gamma \Vdash \pi)^{\bullet}}$$

$$\ldots$$

$$\boxed{(\Gamma \vdash M : \tau)^{\bullet}}$$

$$\ldots$$

Fig. 6. Translating predicates and terms

2.2 Typed translation

$$\boxed{\Gamma \vdash \tau \rightsquigarrow \upsilon : \kappa}$$

$$(\textsc{c-foo})\ \frac{A}{B}$$

$$\boxed{\Gamma \vdash M \rightsquigarrow N : \tau}$$

$$(\textsc{c-foo})\ \frac{A}{B}$$

$$\boxed{\Gamma \Vdash \pi \rightsquigarrow N}$$

$$(\textsc{c-foo})\ \frac{A}{B}$$

$$\boxed{\tau \equiv \upsilon \rightsquigarrow P}$$

$$(\textsc{c-foo})\ \frac{A}{B}$$

Fig. 7. Translation of Rω derivations to Ix derivations

2.3 Properties of Translation

Presume an Rω instantiation of the simple row theory. A lot of this is likely bullshit.

THEOREM 1 (TRANSLATIONAL SOUNDNESS (TYPES)). *if* $\Gamma \vdash \tau : \kappa$ *such that* $\Gamma \vdash \tau \rightsquigarrow \upsilon : \kappa$ *then* $(\Gamma)^{\bullet} \vdash \upsilon : (\kappa)^{\bullet}$.

THEOREM 2 (TRANSLATIONAL SOUNDNESS (TYPE EQUIVALENCE)). *if*

(1) $\Gamma \vdash \tau_1 \rightsquigarrow \upsilon_1 : \kappa_1$;
(2) $\Gamma \vdash \tau_2 \rightsquigarrow \upsilon_2 : \kappa_2$; *and*
(3) $\tau_1 \equiv \tau_2 \rightsquigarrow P$,

then $(\Gamma)^{\bullet} \vdash P : \upsilon_1 \equiv \upsilon_2$.

THEOREM 3 (TRANSLATIONAL SOUNDNESS (OF PREDICATES)). *if* $\Gamma \Vdash \pi$ *such that* $\Gamma \Vdash \pi \rightsquigarrow N$ *then* $(\Gamma)^{\bullet} \vdash N : (\pi)^{\bullet}$.

Finally,

Theorem 4 (Translational Soundness). *if* $\Gamma \vdash M : \tau$ *such that* $\Gamma \vdash M \rightsquigarrow N : \tau$ *then* $(\Gamma)^{\bullet} \vdash N : (\tau)^{\bullet}$.

3 OPERATIONAL SEMANTICS

REFERENCES

Andreas Abel, Joakim Öhman, and Andrea Vezzosi. 2018. Decidability of conversion for type theory in type theory. *Proc. ACM Program. Lang.* 2, POPL (2018), 23:1–23:29. https://doi.org/10.1145/3158111

Alex Hubers and J. Garrett Morris. 2023. Generic Programming with Extensible Data Types; Or, Making Ad Hoc Extensible Data Types Less Ad Hoc. *CoRR* abs/2307.08759 (2023). https://doi.org/10.48550/arXiv.2307.08759 arXiv:2307.08759

J. Garrett Morris and James McKinna. 2019. Abstracting extensible data types: or, rows by any other name. *Proc. ACM Program. Lang.* 3, POPL (2019), 12:1–12:28. https://doi.org/10.1145/3290325