

# Extraction from classical proofs with uniformities (a case study)

Trifon Trifonov

Ludwig Maximilian Universität, München

Russell'08 Workshop  
Swansea, 16 March 2008

# Gödel's T

## Types

$$\sigma, \rho ::= \text{nat} \mid \text{bool} \mid \mathbb{L}(\rho) \mid \sigma \Rightarrow \rho \mid \sigma \times \rho$$

## Terms

# Gödel's T

## Types

$$\sigma, \rho ::= \text{nat} \mid \text{bool} \mid \mathbb{L}(\rho) \mid \sigma \Rightarrow \rho \mid \sigma \times \rho$$

## Terms

$$t, s ::= x \mid ts \mid \lambda_x t \mid \langle t, s \rangle \mid t_{\perp} \mid t_{\lrcorner}$$

# Gödel's T

## Types

$$\sigma, \rho ::= \text{nat} \mid \text{bool} \mid \mathbb{L}(\rho) \mid \sigma \Rightarrow \rho \mid \sigma \times \rho$$

## Terms

$$t, s ::= x \mid ts \mid \lambda_x t \mid \langle t, s \rangle \mid t_{\perp} \mid t_{\lrcorner} \mid$$

$$\text{tt}^{\text{bool}} \mid \text{ff}^{\text{bool}} \mid 0^{\text{nat}} \mid \mathbf{S}^{\text{nat} \Rightarrow \text{nat}} \mid \text{nil}^{\mathbb{L}(\rho)} \mid x^{\rho} ::= f^{\mathbb{L}(\rho)}$$

# Gödel's T

## Types

$$\sigma, \rho ::= \text{nat} \mid \text{bool} \mid \mathbb{L}(\rho) \mid \sigma \Rightarrow \rho \mid \sigma \times \rho$$

## Terms

$$\begin{aligned}
 t, s ::= & x \mid ts \mid \lambda_x t \mid \langle t, s \rangle \mid t_{\perp} \mid t_{\lrcorner} \mid \\
 & \text{tt}^{\text{bool}} \mid \text{ff}^{\text{bool}} \mid 0^{\text{nat}} \mid \mathbf{S}^{\text{nat} \Rightarrow \text{nat}} \mid \text{nil}^{\mathbb{L}(\rho)} \mid x^\rho :: I^{\mathbb{L}(\rho)} \\
 & \text{if } b^{\text{bool}} \text{ then } s^\tau \text{ else } t^\tau \mid
 \end{aligned}$$

# Gödel's T

## Types

$$\sigma, \rho ::= \text{nat} \mid \text{bool} \mid \mathbb{L}(\rho) \mid \sigma \Rightarrow \rho \mid \sigma \times \rho$$

## Terms

$$\begin{aligned}
 t, s & ::= x \mid ts \mid \lambda_x t \mid \langle t, s \rangle \mid t_{\perp} \mid t_{\sqcup} \mid \\
 & \text{tt}^{\text{bool}} \mid \text{ff}^{\text{bool}} \mid 0^{\text{nat}} \mid \mathbf{S}^{\text{nat} \Rightarrow \text{nat}} \mid \text{nil}^{\mathbb{L}(\rho)} \mid x^\rho :: I^{\mathbb{L}(\rho)} \\
 & \mathbf{if} \, b^{\text{bool}} \, \mathbf{then} \, s^\tau \, \mathbf{else} \, t^\tau \mid \\
 & \mathcal{R}_{\text{nat}}^\tau : \text{nat} \Rightarrow \tau \Rightarrow (\text{nat} \Rightarrow \tau \Rightarrow \tau) \Rightarrow \tau \mid
 \end{aligned}$$

# Gödel's T

## Types

$$\sigma, \rho ::= \text{nat} \mid \text{bool} \mid \mathbb{L}(\rho) \mid \sigma \Rightarrow \rho \mid \sigma \times \rho$$

## Terms

$$\begin{aligned}
 t, s & ::= x \mid ts \mid \lambda_x t \mid \langle t, s \rangle \mid t_{\perp} \mid t_{\lrcorner} \mid \\
 & \text{tt}^{\text{bool}} \mid \text{ff}^{\text{bool}} \mid 0^{\text{nat}} \mid \mathbf{S}^{\text{nat} \Rightarrow \text{nat}} \mid \text{nil}^{\mathbb{L}(\rho)} \mid x^\rho :: I^{\mathbb{L}(\rho)} \\
 & \text{if } b^{\text{bool}} \text{ then } s^\tau \text{ else } t^\tau \mid \\
 \mathcal{R}_{\text{nat}}^\tau & : \text{nat} \Rightarrow \tau \Rightarrow (\text{nat} \Rightarrow \tau \Rightarrow \tau) \Rightarrow \tau \mid \\
 \mathcal{R}_{\mathbb{L}(\rho)}^\tau & : \mathbb{L}(\rho) \Rightarrow \tau \Rightarrow (\rho \Rightarrow \mathbb{L}(\rho) \Rightarrow \tau \Rightarrow \tau) \Rightarrow \tau
 \end{aligned}$$

## Negative Arithmetic ( $\text{NA}^\omega$ ) — formulas

We consider the negative fragment of Heyting Arithmetic.

$$A, B ::= P(\vec{t}) \mid \text{at}(b^{\text{bool}}) \mid A \rightarrow B \mid A \wedge B \mid \forall_x A$$

Here  $\perp$  is just a nullary predicate variable.

## Negative Arithmetic ( $\text{NA}^\omega$ ) — formulas

We consider the negative fragment of Heyting Arithmetic.

$$\begin{aligned} A, B &::= P(\vec{t}) \mid \text{at}(b^{\text{bool}}) \mid A \rightarrow B \mid A \wedge B \mid \forall_x A \\ \neg A &::= A \rightarrow \perp \end{aligned}$$

Here  $\perp$  is just a nullary predicate variable.

## Negative Arithmetic ( $\text{NA}^\omega$ ) — formulas

We consider the negative fragment of Heyting Arithmetic.

$$A, B ::= P(\vec{t}) \mid \text{at}(b^{\text{bool}}) \mid A \rightarrow B \mid A \wedge B \mid \forall_x A$$

$$\neg A ::= A \rightarrow \perp$$

$$\tilde{\exists}_x A ::= \neg \forall_x \neg A$$

Here  $\perp$  is just a nullary predicate variable.

## Negative Arithmetic ( $\text{NA}^\omega$ ) — formulas

We consider the negative fragment of Heyting Arithmetic.

$$A, B ::= P(\vec{t}) \mid \text{at}(b^{\text{bool}}) \mid A \rightarrow B \mid A \wedge B \mid \forall_x A$$

$$\neg A ::= A \rightarrow \perp$$

$$\tilde{\exists}_x A ::= \neg \forall_x \neg A$$

Here  $\perp$  is just a nullary predicate variable.

## Negative Arithmetic ( $\text{NA}^\omega$ ) — proofs

$$M, N ::= u^A \mid (\lambda_{u^A} M^B)^{A \rightarrow B} \mid (M^{A \rightarrow B} N^A)^B \mid$$



## Negative Arithmetic ( $\text{NA}^\omega$ ) — proofs

$$M, N ::= u^A \mid (\lambda_{u^A} M^B)^{A \rightarrow B} \mid (M^{A \rightarrow B} N^A)^B \mid \\ (\lambda_x M^A)^{\forall_x A} \text{ (v.c)} \mid (M^{\forall_x A} r)^{A[x:=r]} \mid$$



## Negative Arithmetic ( $\text{NA}^\omega$ ) — proofs

$$\begin{aligned}
 M, N \quad ::= & \quad u^A \mid (\lambda_{u^A} M^B)^{A \rightarrow B} \mid (M^{A \rightarrow B} N^A)^B \mid \\
 & \quad (\lambda_x M^A)^{\forall_x A} \text{ (v.c)} \mid (M^{\forall_x A} r)^{A[x:=r]} \mid \\
 & \quad \langle M^A, N^B \rangle^{A \wedge B} \mid (M^{A \wedge B} \_ \_)^A \mid (M^{A \wedge B} \_ \_)^B \mid
 \end{aligned}$$

## Negative Arithmetic ( $\text{NA}^\omega$ ) — proofs

$$\begin{aligned}
 M, N \quad ::= & \quad u^A \mid (\lambda_{u^A} M^B)^{A \rightarrow B} \mid (M^{A \rightarrow B} N^A)^B \mid \\
 & \quad (\lambda_x M^A)^{\forall_x A} \text{ (v.c)} \mid (M^{\forall_x A} r)^{A[x:=r]} \mid \\
 & \quad \langle M^A, N^B \rangle^{A \wedge B} \mid (M^{A \wedge B} \_ \_ )^A \mid (M^{A \wedge B} \_ \_ )^B \mid \\
 & \quad \text{Cases}_{b,A} : \forall_b (A(\text{ff}) \rightarrow A(\text{tt}) \rightarrow A(b)) \mid
 \end{aligned}$$

## Negative Arithmetic ( $\text{NA}^\omega$ ) — proofs

$$\begin{aligned}
 M, N \quad ::= & \quad u^A \mid (\lambda_{u^A} M^B)^{A \rightarrow B} \mid (M^{A \rightarrow B} N^A)^B \mid \\
 & \quad (\lambda_x M^A)^{\forall_x A} \text{ (v.c)} \mid (M^{\forall_x A} r)^{A[x:=r]} \mid \\
 & \quad \langle M^A, N^B \rangle^{A \wedge B} \mid (M^{A \wedge B} \_ \_)^A \mid (M^{A \wedge B} \_ \_)^B \mid \\
 & \quad \text{Cases}_{b,A} : \forall_b (A(\text{ff}) \rightarrow A(\text{tt}) \rightarrow A(b)) \mid \\
 & \quad \text{Ind}_{n,A} : \forall_n (A(0) \rightarrow \forall_n (A(n) \rightarrow A(Sn)) \rightarrow A(n)) \mid
 \end{aligned}$$

## Negative Arithmetic ( $\text{NA}^\omega$ ) — proofs

$$\begin{aligned}
 M, N \quad ::= & \quad u^A \mid (\lambda_{u^A} M^B)^{A \rightarrow B} \mid (M^{A \rightarrow B} N^A)^B \mid \\
 & \quad (\lambda_x M^A)^{\forall_x A} \text{ (v.c)} \mid (M^{\forall_x A} r)^{A[x:=r]} \mid \\
 & \quad \langle M^A, N^B \rangle^{A \wedge B} \mid (M^{A \wedge B} \_ \_)^A \mid (M^{A \wedge B} \_ \_)^B \mid \\
 & \quad \text{Cases}_{b,A} : \forall_b (A(\text{ff}) \rightarrow A(\text{tt}) \rightarrow A(b)) \mid \\
 & \quad \text{Ind}_{n,A} : \forall_n (A(0) \rightarrow \forall_n (A(n) \rightarrow A(Sn)) \rightarrow A(n)) \mid \\
 & \quad \text{Ind}_{l,A} : \forall_l (A(\text{nil}) \rightarrow \forall_{x,l} (A(l) \rightarrow A(x :: l)) \rightarrow A(l)) \mid
 \end{aligned}$$

## Negative Arithmetic ( $\text{NA}^\omega$ ) — proofs

$$\begin{aligned}
 M, N \quad ::= & \quad u^A \mid (\lambda_{u^A} M^B)^{A \rightarrow B} \mid (M^{A \rightarrow B} N^A)^B \mid \\
 & \quad (\lambda_x M^A)^{\forall_x A} \text{ (v.c)} \mid (M^{\forall_x A} r)^{A[x:=r]} \mid \\
 & \quad \langle M^A, N^B \rangle^{A \wedge B} \mid (M^{A \wedge B} \_ \_)^A \mid (M^{A \wedge B} \_ \_)^B \mid \\
 \text{Cases}_{b,A} : & \quad \forall_b (A(\text{ff}) \rightarrow A(\text{tt}) \rightarrow A(b)) \mid \\
 \text{Ind}_{n,A} : & \quad \forall_n (A(0) \rightarrow \forall_n (A(n) \rightarrow A(Sn)) \rightarrow A(n)) \mid \\
 \text{Ind}_{l,A} : & \quad \forall_l (A(\text{nil}) \rightarrow \forall_{x,l} (A(l) \rightarrow A(x :: l)) \rightarrow A(l)) \mid \\
 \text{Truth} : & \quad \text{at}(\text{tt})
 \end{aligned}$$

## Notation shortcuts

We will use a special nulltype symbol  $\varepsilon$  and stipulate:

$$\begin{array}{lll}
 \varepsilon \times \rho \rightsquigarrow \rho, & t_{\perp} \rightsquigarrow t, & \langle t, \varepsilon \rangle \rightsquigarrow t \\
 \rho \times \varepsilon \rightsquigarrow \rho, & t_{\lrcorner} \rightsquigarrow t, & \langle \varepsilon, t \rangle \rightsquigarrow t \\
 \rho \Rightarrow \varepsilon \rightsquigarrow \varepsilon, & \lambda_x \varepsilon \rightsquigarrow \varepsilon, & \varepsilon t \rightsquigarrow \varepsilon \\
 \varepsilon \Rightarrow \rho \rightsquigarrow \rho, & \lambda_x \varepsilon t \rightsquigarrow t, & t \varepsilon \rightsquigarrow t \\
 & \forall_x \varepsilon \mathbf{A} \rightsquigarrow \mathbf{A}, & M \varepsilon \rightsquigarrow M
 \end{array}$$

We could have used a `unit` type, but then the equalities above become explicit isomorphisms.

## Boolean falsity

Using a general predicate variable  $\perp$  we work in a minimal logic setting.

However, if we use *decidable falsity*  $F := \text{at}(\text{ff})$ , we are able to prove by induction on the definition of formulas

Lemma (ex falso quodlibet)

$$F \rightarrow A$$

Lemma (stability)

$$((A \rightarrow F) \rightarrow F) \rightarrow A$$

if  $A$  contains no predicate variables.

## Boolean falsity

Using a general predicate variable  $\perp$  we work in a minimal logic setting.

However, if we use *decidable falsity*  $F := \text{at}(\text{ff})$ , we are able to prove by induction on the definition of formulas

Lemma (ex falso quodlibet)

$$F \rightarrow A$$

Lemma (stability)

$$((A \rightarrow F) \rightarrow F) \rightarrow A$$

if  $A$  contains no predicate variables.

## Boolean falsity

Using a general predicate variable  $\perp$  we work in a minimal logic setting.

However, if we use *decidable falsity*  $F := \text{at}(\text{ff})$ , we are able to prove by induction on the definition of formulas

**Lemma (ex falso quodlibet)**

$$F \rightarrow A$$

**Lemma (stability)**

$$((A \rightarrow F) \rightarrow F) \rightarrow A$$

if  $A$  contains no predicate variables.

## Boolean falsity

Using a general predicate variable  $\perp$  we work in a minimal logic setting.

However, if we use *decidable falsity*  $F := \text{at}(\text{ff})$ , we are able to prove by induction on the definition of formulas

**Lemma (ex falso quodlibet)**

$$F \rightarrow A$$

**Lemma (stability)**

$$((A \rightarrow F) \rightarrow F) \rightarrow A$$

if  $A$  contains no predicate variables.

## Boolean falsity

Using a general predicate variable  $\perp$  we work in a minimal logic setting.

However, if we use *decidable falsity*  $F := \text{at}(\text{ff})$ , we are able to prove by induction on the definition of formulas

**Lemma (ex falso quodlibet)**

$$F \rightarrow A$$

**Lemma (stability)**

$$((A \rightarrow F) \rightarrow F) \rightarrow A$$

if  $A$  contains no predicate variables.

# Realisability and Dialectica

## Realisability

- ▶ Formulas  $A$  are problems
- ▶ They ask for solutions of type  $A^\circ$
- ▶ Translations  $|A|^r$  verify if  $r$  is a solution to  $A$

## Dialectica

- ▶ Formulas  $A$  are problems
- ▶ They ask for solutions of type  $A^+$
- ▶ that are challenged by terms of type  $A^-$
- ▶ Translations  $|A|_s^r$  verify if  $r$  solves  $A$  for a challenge  $s$

# Realisability and Dialectica

## Realisability

- ▶ Formulas  $A$  are problems
- ▶ They ask for solutions of type  $A^\circ$
- ▶ Translations  $|A|^r$  verify if  $r$  is a solution to  $A$

## Dialectica

- ▶ Formulas  $A$  are problems
- ▶ They ask for solutions of type  $A^+$
- ▶ that are challenged by terms of type  $A^-$
- ▶ Translations  $|A|_s^r$  verify if  $r$  solves  $A$  for a challenge  $s$

# Computational type

$C$	$C^\circ$	$C^+$	$C^-$
$P(\vec{t})$	$\tau$	$\tau^+$	$\tau^-$
$\text{at}(b)$	$\varepsilon$	$\varepsilon$	$\varepsilon$
$A \wedge B$	$A^\circ \times B^\circ$	$A^+ \times B^+$	$A^- \times B^-$
$A \rightarrow B$	$A^\circ \Rightarrow B^\circ$	$(A^+ \Rightarrow B^+) \times$ $(A^+ \Rightarrow B^- \Rightarrow A^-)$	$A^+ \times B^-$
$\forall_{X^p} A$	$\rho \Rightarrow A^\circ$	$\rho \Rightarrow A^+$	$\rho \times A^-$

Realisability needs predicates with computational content.

# Computational type

$C$	$C^\circ$	$C^+$	$C^-$
$P(\vec{t})$	$\tau$	$\tau^+$	$\tau^-$
$\text{at}(b)$	$\varepsilon$	$\varepsilon$	$\varepsilon$
$A \wedge B$	$A^\circ \times B^\circ$	$A^+ \times B^+$	$A^- \times B^-$
$A \rightarrow B$	$A^\circ \Rightarrow B^\circ$	$(A^+ \Rightarrow B^+) \times$ $(A^+ \Rightarrow B^- \Rightarrow A^-)$	$A^+ \times B^-$
$\forall_{X\rho} A$	$\rho \Rightarrow A^\circ$	$\rho \Rightarrow A^+$	$\rho \times A^-$

Realisability needs predicates with computational content.

# Translation

$C$	$ C ^r$	$ C _s^r$
$P(\vec{t})$	$P^\circ(r^{\tau^\circ}, \vec{t})$	$P^\pm(r^{\tau^+}, s^{\tau^-}, \vec{t})$
$\text{at}(b)$	$\text{at}(b)$	$\text{at}(b)$
$A \wedge B$	$ A ^{r_L} \wedge  B ^{r_\perp}$	$ A _{s_L}^{r_L} \wedge  B _{s_\perp}^{r_\perp}$
$A \rightarrow B$	$\forall x ( A ^x \rightarrow  B ^{rx})$	$ A _{(r_\perp)(s_L)(s_\perp)}^{s_L} \rightarrow  B _{s_\perp}^{(r_L)(s_L)}$
$\forall x A(x)$	$\forall x  A(x) ^{rx}$	$ A(s_L) _{s_\perp}^{r(s_L)}$

The Dialectica translation is always a quantifier-free formula.

# Translation

$C$	$ C ^r$	$ C _s^r$
$P(\vec{t})$	$P^\circ(r^{\tau^\circ}, \vec{t})$	$P^\pm(r^{\tau^+}, s^{\tau^-}, \vec{t})$
$\text{at}(b)$	$\text{at}(b)$	$\text{at}(b)$
$A \wedge B$	$ A ^{r_L} \wedge  B ^{r_\perp}$	$ A _{s_L}^{r_L} \wedge  B _{s_\perp}^{r_\perp}$
$A \rightarrow B$	$\forall x ( A ^x \rightarrow  B ^{rx})$	$ A _{(r_\perp)(s_L)(s_\perp)}^{s_L} \rightarrow  B _{s_\perp}^{(r_L)(s_L)}$
$\forall x A(x)$	$\forall x  A(x) ^{rx}$	$ A(s_L) _{s_\perp}^{r(s_L)}$

The Dialectica translation is always a quantifier-free formula.

# Soundness

## Theorem (Realisability)

Let  $\mathcal{P}$  be a proof of  $A$  from assumptions  $u_i : C_i$ . Then we can prove  $|A|_{\llbracket \mathcal{P} \rrbracket^{\circ}}$  from assumptions  $|C_i|^{x_{u_i}}$ .

## Theorem (Dialectica)

Let  $\mathcal{P}$  be a proof of  $A$  from assumptions  $u_i : C_i$ . Then we can prove  $|A|_{y_A}^{\llbracket \mathcal{P} \rrbracket^+}$  from assumptions  $|C_i|_{\llbracket \mathcal{P} \rrbracket_i^-}^{x_{u_i}}$  and  $y_A \notin FV(\llbracket \mathcal{P} \rrbracket^+)$ .

# Soundness

## Theorem (Realisability)

Let  $\mathcal{P}$  be a proof of  $A$  from assumptions  $u_i : C_i$ . Then we can prove  $|A|_{\llbracket \mathcal{P} \rrbracket^{\circ}}$  from assumptions  $|C_i|^{x_{u_i}}$ .

## Theorem (Dialectica)

Let  $\mathcal{P}$  be a proof of  $A$  from assumptions  $u_i : C_i$ . Then we can prove  $|A|_{y_A}^{\llbracket \mathcal{P} \rrbracket^+}$  from assumptions  $|C_i|_{\llbracket \mathcal{P} \rrbracket_i^-}^{x_{u_i}}$  and  $y_A \notin FV(\llbracket \mathcal{P} \rrbracket^+)$ .







## Extracted terms (ctd.)

$\mathcal{P}$	$\llbracket \mathcal{P} \rrbracket^\circ$	$\llbracket \mathcal{P} \rrbracket^+$	$\llbracket \mathcal{P} \rrbracket_i^-$
$\lambda_u M$	$\lambda_{x_u} \llbracket M \rrbracket^\circ$	$\langle \lambda_{x_u} \llbracket M \rrbracket^+, \lambda_{x_u, y_B} \llbracket M \rrbracket_u^- \rangle$	$\llbracket M \rrbracket_i^- [x_u := y_{A \rightarrow B \perp}] [y_B := y_{A \rightarrow B \neg}]$
$MN$	$\llbracket M \rrbracket^\circ \llbracket N \rrbracket^\circ$	$(\llbracket M \rrbracket^+ \perp) \llbracket N \rrbracket^+$	$\llbracket M \rrbracket_i^- [y_{A \rightarrow B} := \langle \llbracket N \rrbracket^+, y_B \rangle]$ $\overset{i}{\bowtie}$ $\llbracket N \rrbracket_i^- [y_A := (\llbracket M \rrbracket^+ \perp) \llbracket N \rrbracket^+ y_B]$

## Extracted terms (ctd.)

$\mathcal{P}$	$[[\mathcal{P}]]^\circ$	$[[\mathcal{P}]]^+$	$[[\mathcal{P}]]_i^-$
$\lambda_x M$	$\lambda_x [[M]]^\circ$	$\lambda_x [[M]]^+$	$[[M]]_i^- [x := y_{\forall_x A}] [y_A := y_{\forall_x A}]$
$Mt$	$[[M]]^\circ t$	$[[M]]^+ t$	$[[M]]_i^- [y_{\forall_x A} := \langle t, y_A \rangle]$

# Realising boolean induction

$\mathcal{P}$	Cases $_{b,A} b M N$
$[[\mathcal{P}]]^\circ$	<b>if <math>b</math> then <math>[[M]]^\circ</math> else <math>[[N]]^\circ</math></b>
$[[\mathcal{P}]]^+$	<b>if <math>b</math> then <math>[[M]]^+</math> else <math>[[N]]^+</math></b>
$[[\mathcal{P}]]_i^-$	$[[M]]_i^- \overset{i}{\bowtie} [[N]]_i^-$



## Realising list induction

$\mathcal{P}$	$\text{Ind}_{l,A} n M N$
$[[\mathcal{P}]]^\circ$	$\mathcal{R}_{L(\rho)} / [[M]]^\circ [[N]]^\circ$
$[[\mathcal{P}]]^+$	$\mathcal{R}_{L(\rho)} / [[M]]^+ (\lambda_{x,l} [[N]]^+ x / \perp)$
$[[\mathcal{P}]]_i^-$	$\mathcal{R}_{L(\rho)} / (\lambda_{y_A} [[M]]_i^-$ $\left( \begin{array}{c} [[N]]_i^- [y_S := \langle x, l, [[\mathcal{P}]]^+ l, y \rangle] \\ \lambda_{x,l,p,y} \quad \quad \quad \begin{array}{c} i \\ \boxtimes \end{array} \\ p([[M]]^+ x / \perp ([[ \mathcal{P} ]]^+ l) y) \end{array} \right)$

## A-translation

### Theorem (Extraction via A-translation)

Let  $\mathcal{P}$  be a proof of

$$\mathbf{NA}^\omega \vdash \vec{D} \rightarrow \exists y^\rho G$$

with  $\vec{D}$  and  $G$  not containing  $\perp$ . Then

$$\mathbf{NA}^\omega \vdash \vec{D} \rightarrow G(\llbracket \mathcal{P} \rrbracket^\circ(\lambda y y)).$$

**Proof.**

Let  $\tau^\circ(\perp) = \rho$  and  $\perp$  be translated to a unary predicate variable  $\mathcal{A}$ . Substitute  $\mathcal{A}$  with  $G(y)$ . □

# A-translation

## Theorem (Extraction via A-translation)

Let  $\mathcal{P}$  be a proof of

$$\mathbf{NA}^\omega \vdash \vec{D} \rightarrow (\forall_{y\rho}. G \rightarrow \perp) \rightarrow \perp$$

with  $\vec{D}$  and  $G$  not containing  $\perp$ . Then

$$\mathbf{NA}^\omega \vdash \vec{D} \rightarrow G([\mathcal{P}]^\circ(\lambda_y y)).$$

**Proof.**

Let  $\tau^\circ(\perp) = \rho$  and  $\perp$  be translated to a unary predicate variable  $\mathcal{A}$ . Substitute  $\mathcal{A}$  with  $G(y)$ . □

## A-translation

### Theorem (Extraction via A-translation)

Let  $\mathcal{P}$  be a proof of

$$\mathbf{NA}^\omega \vdash \vec{D} \rightarrow (\forall_{y^\rho}. G \rightarrow \perp) \rightarrow \perp$$

with  $\vec{D}$  and  $G$  not containing  $\perp$ . Then

$$\mathbf{NA}^\omega \vdash \vec{D} \rightarrow G([\mathcal{P}]^\circ(\lambda_y y)).$$

**Proof.**

Let  $\tau^\circ(\perp) = \rho$  and  $\perp$  be translated to a unary predicate variable  $\mathcal{A}$ . Substitute  $\mathcal{A}$  with  $G(y)$ . □

## A-translation

### Theorem (Extraction via A-translation)

Let  $\mathcal{P}$  be a proof of

$$\mathbf{NA}^\omega \vdash \vec{D} \rightarrow (\forall_{y^\rho}. G \rightarrow \perp) \rightarrow \perp$$

with  $\vec{D}$  and  $G$  not containing  $\perp$ . Then

$$\mathbf{NA}^\omega \vdash \vec{D} \rightarrow G(\llbracket \mathcal{P} \rrbracket^\circ(\lambda_y y)).$$

**Proof.**

Let  $\tau^\circ(\perp) = \rho$  and  $\perp$  be translated to a unary predicate variable  $\mathcal{A}$ . Substitute  $\mathcal{A}$  with  $G(y)$ . □

# A-translation

## Theorem (Extraction via A-translation)

Let  $\mathcal{P}$  be a proof of

$$\text{NA}^\omega \vdash \vec{D} \rightarrow (\forall_{y^\rho}. G \rightarrow \perp) \rightarrow \perp$$

with  $\vec{D}$  and  $G$  not containing  $\perp$ . Then

$$\text{NA}^\omega \vdash \vec{D} \rightarrow G(\llbracket \mathcal{P} \rrbracket^\circ(\lambda_y y)).$$

### Proof.

Let  $\tau^\circ(\perp) = \rho$  and  $\perp$  be translated to a unary predicate variable  $\mathcal{A}$ . Substitute  $\mathcal{A}$  with  $G(y)$ . □

## Uniform quantifiers

Consider the formula

$$\forall_n \forall_m (n = 2m \rightarrow \exists_k n^2 = 4k)$$

A realiser could be  $k = m^2$  or  $k = \text{Quot}(n^2, 4)$ . Hence, a quantified variable is not necessarily computationally relevant. If not, we quantify it *uniformly*:  $\forall_x^U A$ , i.e., the verifying proof is *uniform* in this variable.

## Uniform quantifiers

Consider the formula

$$\forall_n \forall_m (n = 2m \rightarrow \exists_k n^2 = 4k)$$

A realiser could be  $k = m^2$  or  $k = \text{Quot}(n^2, 4)$ . Hence, a quantified variable is not necessarily computationally relevant. If not, we quantify it *uniformly*:  $\forall_x^U A$ , i.e., the verifying proof is *uniform* in this variable.

## Uniform quantifiers

Consider the formula

$$\forall_n \forall_m (n = 2m \rightarrow \exists_k n^2 = 4k)$$

A realiser could be  $k = m^2$  or  $k = \text{Quot}(n^2, 4)$ . Hence, a quantified variable is not necessarily computationally relevant. If not, we quantify it *uniformly*:  $\forall_x^U A$ , i.e., the verifying proof is *uniform* in this variable.

## Uniform quantifiers

Consider the formula

$$\forall_n \forall_m (n = 2m \rightarrow \exists_k n^2 = 4k)$$

A realiser could be  $k = m^2$  or  $k = \text{Quot}(n^2, 4)$ . Hence, a quantified variable is not necessarily computationally relevant.

If not, we quantify it *uniformly*:  $\forall_x^U A$ , i.e., the verifying proof is *uniform* in this variable.

## Uniform quantifiers

Consider the formula

$$\forall_n \forall_m (n = 2m \rightarrow \exists_k n^2 = 4k)$$

A realiser could be  $k = m^2$  or  $k = \text{Quot}(n^2, 4)$ . Hence, a quantified variable is not necessarily computationally relevant. If not, we quantify it *uniformly*:  $\forall_x^U A$ , i.e., the verifying proof is *uniform* in this variable.

## Uniform quantifiers

Consider the formula

$$\forall_n \forall_m (n = 2m \rightarrow \exists_k n^2 = 4k)$$

A realiser could be  $k = m^2$  or  $k = \text{Quot}(n^2, 4)$ . Hence, a quantified variable is not necessarily computationally relevant. If not, we quantify it *uniformly*:  $\forall_x^U A$ , i.e., the verifying proof is *uniform* in this variable.

## Negative Arithmetic ( $\text{NA}^\omega$ ) — formulas

We consider the negative fragment of Heyting Arithmetic.

$$A, B ::= P(\vec{t}) \mid \text{at}(b^{\text{bool}}) \mid A \rightarrow B \mid A \wedge B \mid \forall_x A \mid \forall_x^U A$$

$$\neg A ::= A \rightarrow \perp$$

$$\tilde{\exists}_x A ::= \neg \forall_x \neg A$$

## Negative Arithmetic ( $\text{NA}^\omega$ ) — proofs

$$\begin{aligned}
 M, N \quad ::= & \quad u^A \mid (\lambda_{u^A} M^B)^{A \rightarrow B} \mid (M^{A \rightarrow B} N^A)^B \mid \\
 & \quad (\lambda_x M^A)^{\forall_x A} \text{ (v.c) } \mid (M^{\forall_x A} r)^{A[x:=r]} \mid \\
 & \quad (\lambda_x^U M^A)^{\forall_x^U A} \text{ (v.c) } \mid (M^{\forall_x^U A} r)^{A[x:=r]} \mid \\
 & \quad \langle M^A, N^B \rangle^{A \wedge B} \mid (M^{A \wedge B} \_ \_ )^A \mid (M^{A \wedge B} \_ \_ )^B \mid \\
 \text{Cases}_{b,A} : & \quad \forall_b (A(\text{ff}) \rightarrow A(\text{tt}) \rightarrow A(b)) \mid \\
 \text{Ind}_{n,A} : & \quad \forall_n (A(0) \rightarrow \forall_n (A(n) \rightarrow A(Sn)) \rightarrow A(n)) \mid \\
 \text{Ind}_{l,A} : & \quad \forall_l (A(\text{nil}) \rightarrow \forall_{x,l} (A(l) \rightarrow A(x :: l)) \rightarrow A(l)) \mid \\
 \text{Truth} : & \quad \text{at}(\text{tt})
 \end{aligned}$$

## Computational type

$C$	$C^\circ$	$C^+$	$C^-$
$P(\vec{t})$	$\tau$	$\tau^+$	$\tau^-$
$\text{at}(b)$	$\varepsilon$	$\varepsilon$	$\varepsilon$
$A \wedge B$	$A^\circ \times B^\circ$	$A^+ \times B^+$	$A^- \times B^-$
$A \rightarrow B$	$A^\circ \Rightarrow B^\circ$	$(A^+ \Rightarrow B^+) \times$ $(A^+ \Rightarrow B^- \Rightarrow A^-)$	$A^+ \times B^-$
$\forall_{x\rho} A$	$\rho \Rightarrow A^\circ$	$\rho \Rightarrow A^+$	$\rho \times A^-$
$\forall_{x\rho}^U A$	$A^\circ$	$A^+$	$A^-$

## Translation

$C$	$ C ^r$	$ C _s^r$
$P(\vec{t})$	$P^o(r^{\tau^o}, \vec{t})$	$P^\pm(r^{\tau^+}, s^{\tau^-}, \vec{t})$
$\text{at}(b)$	$\text{at}(b)$	$\text{at}(b)$
$A \wedge B$	$ A ^{r_L} \wedge  B ^{r_R}$	$ A _{s_L}^{r_L} \wedge  B _{s_R}^{r_R}$
$A \rightarrow B$	$\forall_x ( A ^x \rightarrow  B ^{rx})$	$ A _{(r_\perp)(s_L)(s_R)}^{s_L} \rightarrow  B _{s_J}^{(r_L)(s_L)}$
$\forall_x A(x)$	$\forall_x  A(x) ^{rx}$	$ A(s_L) _{s_J}^{r(s_L)}$
$\forall_x^U A(x)$	$\forall_x  A(x) ^r$	$\forall_x  A(x) _s^r$

The Dialectica translation is **not** always quantifier-free! ▶▶

# Translation

$C$	$ C ^r$	$ C _s^r$
$P(\vec{t})$	$P^\circ(r^{\tau^\circ}, \vec{t})$	$P^\pm(r^{\tau^+}, s^{\tau^-}, \vec{t})$
$\text{at}(b)$	$\text{at}(b)$	$\text{at}(b)$
$A \wedge B$	$ A ^{r_\perp} \wedge  B ^{r_\perp}$	$ A _{s_\perp}^{r_\perp} \wedge  B _{s_\perp}^{r_\perp}$
$A \rightarrow B$	$\forall_x ( A ^x \rightarrow  B ^{rx})$	$ A _{(r_\perp)(s_\perp)}^{s_\perp} \rightarrow  B _{s_\perp}^{(r_\perp)(s_\perp)}$
$\forall_x A(x)$	$\forall_x  A(x) ^{rx}$	$ A(s_\perp) _{s_\perp}^{r(s_\perp)}$
$\forall_x^U A(x)$	$\forall_x  A(x) ^r$	$\forall_x  A(x) _s^r$

The Dialectica translation is **not** always quantifier-free! 

## Extracted terms

$\mathcal{P}$	$[[\mathcal{P}]]^\circ$	$[[\mathcal{P}]]^+$	$[[\mathcal{P}]]_i^-$
$\lambda_x M$	$\lambda_x [[M]]^\circ$	$\lambda_x [[M]]^+$	$[[M]]_i^- [x := y_{\forall_x A}] [y_A := y_{\forall_x A}]$
$Mt$	$[[M]]^\circ t$	$[[M]]^+ t$	$[[M]]_i^- [y_{\forall_x A} := \langle t, y_A \rangle]$
$\lambda_x^U M$	$[[M]]^\circ$	$[[M]]^+$	$[[M]]_i^- [y_A := y_{\forall_x^U A}]$
$M^{\forall_x^U A} t$	$[[M]]^\circ$	$[[M]]^+$	$[[M]]_i^- [y_{\forall_x^U A} := y_A]$

## Uniformisation correctness

### Definition (Realisability)

$\lambda_x^U M$  is correct if  $x \notin FV(\llbracket M \rrbracket^\circ)$

### Definition (Dialectica)

$\lambda_x^U M$  is correct if  $x \notin FV(\llbracket M \rrbracket^+)$  and  $x \notin FV(\llbracket M \rrbracket_i^-)$

$\mathcal{P}$  is correct if also for every  $t_1 \overset{A}{\bowtie} t_2$ ,  $\forall^U$  does not occur in  $A$

# Uniformisation correctness

## Definition (Realisability)

$\lambda_x^U M$  is correct if  $x \notin FV(\llbracket M \rrbracket^\circ)$

## Definition (Dialectica)

$\lambda_x^U M$  is correct if  $x \notin FV(\llbracket M \rrbracket^+)$  and  $x \notin FV(\llbracket M \rrbracket_i^-)$

$\mathcal{P}$  is correct if also for every  $t_1 \overset{A}{\bowtie} t_2$ ,  $\forall^U$  does not occur in  $A$

# Uniformisation correctness

## Definition (Realisability)

$\lambda_x^U M$  is correct if  $x \notin FV(\llbracket M \rrbracket^\circ)$

## Definition (Dialectica)

$\lambda_x^U M$  is correct if  $x \notin FV(\llbracket M \rrbracket^+)$  and  $x \notin FV(\llbracket M \rrbracket_i^-)$

$\mathcal{P}$  is correct if also for every  $t_1 \overset{A}{\bowtie} t_2$ ,  $\forall^U$  does not occur in  $A$

## List reversal

Let  $\text{Rev}$  be a binary predicate without computational content for the graph of the list reversal function. **Assumptions:**

$$R_0 : \quad \text{Rev}(\text{nil}, \text{nil})$$

$$R_1 : \quad \forall x, l_1, l_2 \left( \text{Rev}(l_1, l_2) \rightarrow \text{Rev}(l_1 :+ x :, x :: l_2) \right)$$

Here  $l_1 :+ l_2$  denotes appending  $l_2$  to  $l_1$ . We assume that  $\text{Rev}$  is undecidable.

## List reversal

Let  $\text{Rev}$  be a binary predicate without computational content for the graph of the list reversal function. Assumptions:

$$R_0 : \quad \text{Rev}(\text{nil}, \text{nil})$$

$$R_1 : \quad \forall_{x, l_1, l_2} (\text{Rev}(l_1, l_2) \rightarrow \text{Rev}(l_1 :+ : x :, x :: l_2))$$

Here  $l_1 :+ : l_2$  denotes appending  $l_2$  to  $l_1$ . We assume that  $\text{Rev}$  is undecidable.

## List reversal

Let  $\text{Rev}$  be a binary predicate without computational content for the graph of the list reversal function. Assumptions:

$$R_0 : \quad \text{Rev}(\text{nil}, \text{nil})$$

$$R_1 : \quad \forall_{x, l_1, l_2} (\text{Rev}(l_1, l_2) \rightarrow \text{Rev}(l_1 :+ : x :, x :: l_2))$$

Here  $l_1 :+ : l_2$  denotes appending  $l_2$  to  $l_1$ . We assume that  $\text{Rev}$  is undecidable.

## List reversal

Let  $\text{Rev}$  be a binary predicate without computational content for the graph of the list reversal function. Assumptions:

$$R_0 : \quad \text{Rev}(\text{nil}, \text{nil})$$

$$R_1 : \quad \forall_{x, l_1, l_2} (\text{Rev}(l_1, l_2) \rightarrow \text{Rev}(l_1 :+ : x :, x :: l_2))$$

Here  $l_1 :+ : l_2$  denotes appending  $l_2$  to  $l_1$ . We assume that  $\text{Rev}$  is undecidable.

We will prove that it is not possible that a list is not reversible.

$$M : \forall l_0 \exists l \text{Rev}(l_0, l)$$

To this end, we prove by induction that if a list is not reversible then none of its initial segments is:

$$\forall l (\text{Rev}(l_0, l) \rightarrow \perp) \rightarrow \forall l_2 \forall l_1 (l_1 :+ : l_2 = l_0 \rightarrow \forall l (\text{Rev}(l_1, l) \rightarrow \perp))$$

We will prove that it is not possible that a list is not reversible.

$$M : \forall l_0 \exists l \text{Rev}(l_0, l)$$

To this end, we prove by induction that if a list is not reversible then none of its initial segments is:

$$\forall l (\text{Rev}(l_0, l) \rightarrow \perp) \rightarrow \forall l_2 \forall l_1 (l_1 :+ l_2 = l_0 \rightarrow \forall l (\text{Rev}(l_1, l) \rightarrow \perp))$$

# Proof

Induction on  $l_2$  for

$$L : \forall l_2 \forall l_1 (l_1 :+ : l_2 = l_0 \rightarrow \forall l (\text{Rev}(l_1, l) \rightarrow \perp))$$

# Proof

Induction on  $l_2$  for

$$L : \forall l_2 \forall l_1 (l_1 :+ : l_2 = l_0 \rightarrow \forall l (\text{Rev}(l_1, l) \rightarrow \perp))$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

# Proof

Induction on  $l_2$  for

$$L : \forall l_2 \forall l_1 (l_1 :+ : l_2 = l_0 \rightarrow \forall l (\text{Rev}(l_1, l) \rightarrow \perp))$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

# Proof

Induction on  $l_2$  for

$$L : \forall l_2 \forall l_1 (l_1 :+ : l_2 = l_0 \rightarrow \forall l (\text{Rev}(l_1, l) \rightarrow \perp))$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

# Proof

Induction on  $l_2$  for

$$L : \forall l_2 \forall l_1 (l_1 :+ : l_2 = l_0 \rightarrow \forall l (\text{Rev}(l_1, l) \rightarrow \perp))$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1 :+ : x : ) L_E^{(l_1 :+ : x : ) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

# Proof

Induction on  $l_2$  for

$$L : \forall l_2 \forall l_1 (l_1 :+ : l_2 = l_0 \rightarrow \forall l (\text{Rev}(l_1, l) \rightarrow \perp))$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1 :+ : x) L_E^{(l_1 :+ : x) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil}:+:l_0=l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_B \rrbracket^\circ := \lambda_u u$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}:+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_B \rrbracket^\circ := \lambda_u u$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}:+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_B \rrbracket^\circ := \lambda_u u$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}:+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_B \rrbracket^\circ := (\lambda_u u) x_u$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}:+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_B \rrbracket^\circ := x_u$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}::l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::(x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1::x::) L_E^{(l_1::x)::l_2=l_0} (x::l) L_{S1}^{\text{Rev}(l_1::x::, x::l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_B \rrbracket^\circ := \lambda_{l_1} x_u$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}::l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::(x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1::x::) L_E^{(l_1::x)::l_2=l_0} (x::l) L_{S1}^{\text{Rev}(l_1::x::, x::l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_S \rrbracket^\circ := x_p$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil}:+:l_0=l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_S \rrbracket^\circ := x_p(l_1::+:x:)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}::+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x::: l) L_{S1}^{\text{Rev}(l_1::+:x:::, x::: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_S \rrbracket^\circ := x_p(l_1::+:x:)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}::+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x::: l) L_{S1}^{\text{Rev}(l_1::+:x:::, x::: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_S \rrbracket^\circ := x_p(l_1::+:x:)(x::l)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}::+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x::l) L_{S1}^{\text{Rev}(l_1::+:x:, x::l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_S \rrbracket^\circ := x_p(l_1::+:x:)(x::l)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}::+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x::l) L_{S1}^{\text{Rev}(l_1::+:x:, x::l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_S \rrbracket^\circ := x_p(l_1::+:x:)(x::l)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil}::+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x::l) L_{S1}^{\text{Rev}(l_1::+:x:, x::l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_S \rrbracket^\circ := \lambda_l x p (l_1 ::+ x) (x :: l)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil}::+l_0=l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+(x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1 ::+ x) L_E^{(l_1::+x)::+l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1::+x::, x::l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L_S \rrbracket^\circ := \lambda_l x p (l_1 :: + : x : ) (x :: l)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{ nil } M_E^{\text{nil} :: + : l_0 = l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :: + : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :: + : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1 :: + : x : ) L_E^{(l_1 :: + : x : ) :: + : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :: + : x : , x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$[[L_S]]^\circ := \lambda l_1 \lambda l \lambda x \rho (l_1 ::+ x) (x :: l)$$

$$M := \lambda R_0, R_1 \lambda l_0 \lambda u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil}::+l_0=l_0} \text{nil } R_0$$

$$L := \lambda l_2 \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda l_1 \lambda v^{l_1::+\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda x, l_2 \lambda \rho \lambda l_1 \lambda v^{l_1::+(x::l_2)=l_0} \lambda l \lambda w^{\text{Rev}(l_1, l)}$$

$$\rho (l_1 ::+ x) L_E^{(l_1::+x)::+l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1::+x::, x::l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket LS \rrbracket^\circ := \lambda_{x_p} \lambda_{l_1} \lambda_l x_p (l_1 :: x) (x :: l)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil} :: l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :: \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :: (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1 :: x) L_E^{(l_1 :: x) :: l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :: x, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket LS \rrbracket^\circ := \lambda_{x,l_2} \lambda_{x_p} \lambda_{l_1} \lambda_l x_p (l_1 :: x) (x :: l)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L l_0 \text{nil } M_E^{\text{nil} :: l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :: \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x,l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :: (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1 :: x) L_E^{(l_1 :: x) :: l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :: x, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L \rrbracket^\circ := \mathcal{R} \, l_0(\lambda_{l_1} x_u) (\lambda_{x,l_2,x_p,l_1,l} x_p(l_1::+:x:)(x::l))$$

$$M := \lambda_{R_0,R_1} \lambda_{l_0} \lambda_u^{\forall l(\text{Rev}(l_0,l) \rightarrow \perp)} L \, l_0 \text{ nil } M_E^{\text{nil}::+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x,l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1,l)}$$

$$p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x::l) L_{S1}^{\text{Rev}(l_1::+:x:,x::l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket L \rrbracket^\circ := \lambda_{l_2} \mathcal{R} \, l_2 (\lambda_{l_1} x_u) (\lambda_{x, l_2, x_p, l_1, l} x_p (l_1 :+ : x :)(x :: l))$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{ nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p (l_1 :+ : x : ) L_E^{(l_1 :+ : x : ) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \mathcal{R} \, l_0(\lambda_{l_1} x_u) (\lambda_{x,l_2,x_p,l_1,l} x_p(l_1::+:x:)(x :: l))$$

$$M := \lambda_{R_0,R_1} \lambda_{l_0} \lambda_u^{\forall l(\text{Rev}(l_0,l) \rightarrow \perp)} L \, l_0 \, \text{nil} \, M_E^{\text{nil}::+:l_0=\text{nil}} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x,l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1,l)} \\ p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1::+:x:,x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \mathcal{R} \, l_0(\lambda_{l_1} x_u) (\lambda_{x, l_2, x_p, l_1, l} x_p (l_1 :: + : x :)) (x :: l) \text{nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{nil} M_E^{\text{nil} :: + : l_0 = l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :: + : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :: + : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p (l_1 :: + : x :) L_E^{(l_1 :: + : x :) :: + : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :: + : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \mathcal{R} \, l_0(\lambda_{l_1} x_u) (\lambda_{x, l_2, x_p, l_1, l} x_p (l_1 :: +: x:)) (x :: l) \text{nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{nil} M_E^{\text{nil}::+:l_0=l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p (l_1 :: +: x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1::+:x::, x::l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \mathcal{R} \, l_0(\lambda_{l_1} x_u) (\lambda_{x,l_2,x_p,l_1,l} x_p(l_1::+:x:))(x::l) \text{nil nil}$$

$$M := \lambda_{R_0,R_1} \lambda_{l_0} \lambda_u^{\forall l(\text{Rev}(l_0,l) \rightarrow \perp)} L \, l_0 \text{nil } M_E^{\text{nil}::+:l_0=l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat } l_1 \, l_0 \, u$$

$$L_S := \lambda_{x,l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1,l)} \\ p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x::l) L_{S1}^{\text{Rev}(l_1::+:x:,x::l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \mathcal{R} \, l_0(\lambda_{l_1} x_u) (\lambda_{x,l_2,x_p,l_1,l} x_p(l_1::+:x:))(x :: l) \text{nil nil}$$

$$M := \lambda_{R_0,R_1} \lambda_{l_0} \lambda_u^{\forall l(\text{Rev}(l_0,l) \rightarrow \perp)} L \, l_0 \text{nil } M_E^{\text{nil}::+:l_0=l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat } l_1 \, l_0 \, u$$

$$L_S := \lambda_{x,l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1,l)} \\ p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1::+:x:,x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \lambda_{x_u} \mathcal{R} \, l_0 (\lambda_{l_1} x_u) (\lambda_{x, l_2, x_p, l_1, l} x_p (l_1 :+ : x : ) (x :: l)) \text{nil nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p (l_1 :+ : x : ) L_E^{(l_1 :+ : x : ) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x : , x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \lambda_{l_0} \lambda_{x_u} \mathcal{R} \, l_0 (\lambda_{l_1} x_u) (\lambda_{x, l_2, x_p, l_1, l} x_p (l_1 :+ : x :)(x :: l)) \text{nil nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p (l_1 :+ : x :)_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \lambda_{l_0} (\lambda_{x_u} \mathcal{R} \, l_0 (\lambda_{l_1} x_u) (\lambda_{x, l_2, x_p, l_1, l} x_p (l_1 :: x) (x :: l)) \text{nil nil}) (\lambda_l l)$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{nil} \, M_E^{\text{nil}::l_0=l_0} \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::\text{nil}=l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::(x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p (l_1 :: x) L_E^{(l_1::x)::l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1::x::x::l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \lambda_{l_0} \mathcal{R} \, l_0 (\lambda_{l_1, l} l) (\lambda_{x, l_2, x_p, l_1, l} x_p (l_1 :+ : x :)(x :: l)) \text{nil nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p (l_1 :+ : x : ) L_E^{(l_1 :+ : x : ) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \lambda_{l_0} \mathcal{R} \, l_0 (\lambda_{l_1, l} (\lambda_{x, l_2, x_p, l_1, l} x_p (l_1 :: x :: l))) \text{nil nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{nil } M_E^{\text{nil}::l_0=l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::\text{nil}=l_0} \text{Compat } l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::(x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p (l_1 :: x :: l) L_E^{(l_1::x)::l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1::x::l, x::l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\forall_{l_2} \forall_{l_1}^U (l_1 :: + : l_2 = l_0 \rightarrow \forall_l (\text{Rev}(l_1, l) \rightarrow \perp))$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall_l (\text{Rev}(l_0, l) \rightarrow \perp)} L_{l_0} \text{nil} M_E^{\text{nil} :: + : l_0 = l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1}^U \lambda_v^{l_1 :: + : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1}^U \lambda_v^{l_1 :: + : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1 :: + : x ::) L_E^{(l_1 :: + : x ::) :: + : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :: + : x ::, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \lambda_{l_0} \mathcal{R} \, l_0 (\lambda_{l_1, l} (\lambda_{x, l_2, x_p, l_1, l} x_p (l_1 :: x :: l))) \text{nil nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{nil } M_E^{\text{nil} :: l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 L_B L_S$$

$$L_B := \lambda_{l_1}^U \lambda_v^{l_1 :: \text{nil} = l_0} \text{Compat } l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1}^U \lambda_v^{l_1 :: (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p (l_1 :: x :: l) L_E^{(l_1 :: x :: l) :: l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :: x :: l, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \lambda_{l_0} \mathcal{R} \, l_0 (\lambda_{l_1} l) (\lambda_{x, l_2, x_p, l} x_p (x :: l)) \text{ nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{ nil } M_E^{\text{nil}::l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1}^U \lambda_v^{l_1::\text{nil}=l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1}^U \lambda_v^{l_1::(x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p \, (l_1::x:) \, L_E^{(l_1::x)::l_2=l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1::x::x::l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by A-translation

Extracted program:

$$\llbracket M \rrbracket^\circ := \lambda_{l_0} \mathcal{R} \, l_0 (\lambda_l l) (\lambda_{x, l_2, x_p, l} x_p (x :: l)) \text{ nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow \perp)} L \, l_0 \text{ nil } M_E^{\text{nil}:+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1}^U \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1}^U \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p \, (l_1:+:x:) \, L_E^{(l_1:+:x):+:l_2=l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u \forall_l (\text{Rev}(l_0, l) \rightarrow F) L l_0 \text{ nil } M_E^{\text{nil}:+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1:+:x:) L_E^{(l_1:+:x:)+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u \forall_l (\text{Rev}(l_0, l) \rightarrow F) L l_0 \text{ nil } M_E^{\text{nil}:+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1:+:x:) L_E^{(l_1:+:x:)+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u \forall_l (\text{Rev}(l_0, l) \rightarrow F) L l_0 \text{ nil } M_E^{\text{nil}:+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1:+:x:) L_E^{(l_1:+:x:)+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u \forall_l (\text{Rev}(l_0, l) \rightarrow F) L l_0 \text{ nil } M_E^{\text{nil}:+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_B]]_U^- := y$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil} M_E^{\text{nil}:+:l_0=l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_B]]_U^- := y$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil } M_E^{\text{nil}:+:l_0=l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_B]]_U^- := y \perp$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil} M_E^{\text{nil}:+:l_0=l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_B]]_U^- := y \perp \quad [[L_S]]_{R_1}^- := y$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil} M_E^{\text{nil}:+:l_0=l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_B]]_U^- := y \perp \quad [[L_S]]_{R_1}^- := \langle x, l_1, l \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil} M_E^{\text{nil}:+:l_0=l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1:+:x:) L_E^{(l_1:+:x:)+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_S]]_p^- := y$$

$$[[L_B]]_u^- := y \perp \quad [[L_S]]_{R_1}^- := \langle x, l_1, l \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil} M_E^{\text{nil}:+:l_0=l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat} l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p (l_1:+:x:) L_E^{(l_1:+:x):+:l_2=l_0} (x :: l) L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket L_S \rrbracket_p^- := \langle l_1 :: x :: y \rangle$$

$$\llbracket L_B \rrbracket_u^- := y \perp \quad \llbracket L_S \rrbracket_{R_1}^- := \langle x, l_1, l \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{ nil } M_E^{\text{nil} :: l_0 = l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :: \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :: (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1 :: x :: y) L_E^{(l_1 :: x :: y) :: l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :: x :: y, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_S]]_p^- := \langle l_1::+:x:, y \rangle$$

$$[[L_B]]_u^- := y \perp \quad [[L_S]]_{R_1}^- := \langle x, l_1, l \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil} M_E^{\text{nil}::+:l_0=l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat} l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1::+:x:) L_E^{(l_1::+:x)::+: l_2=l_0} (x::: l) L_{S1}^{\text{Rev}(l_1::+:x:, x::: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket L_S \rrbracket_p^- := \langle l_1 :+ : x :, x :: l \rangle$$

$$\llbracket L_B \rrbracket_u^- := y \perp \quad \llbracket L_S \rrbracket_{R_1}^- := \langle x, l_1, l \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1 :+ : x :) L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket L_S \rrbracket_p^- := \langle l_1 :+ : x :, x :: l \rangle$$

$$\llbracket L_B \rrbracket_u^- := y \perp \quad \llbracket L_S \rrbracket_{R_1}^- := \langle x, l_1, l \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1 :+ : x :) L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket L_S \rrbracket_p^- := \langle l_1 :+ : x :, x :: l \rangle$$

$$\llbracket L_B \rrbracket_u^- := y \perp \quad \llbracket L_S \rrbracket_{R_1}^- := \langle x, l_1, l \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1 :+ : x :) L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_S]]_p^- := \langle l_1::+:x:, x::y \rangle$$

$$[[L_B]]_u^- := y \perp \quad [[L_S]]_{R_1}^- := \langle x, l_1, y \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{ nil } M_E^{\text{nil}::+:l_0=l_0} \text{ nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x::l) L_{S1}^{\text{Rev}(l_1::+:x:, x::l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_S]]_p^- := \langle l_1::+:x:, x::y \rangle$$

$$[[L_B]]_u^- := y \perp \quad [[L_S]]_{R_1}^- := \langle x, l_1, y \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil } M_E^{\text{nil}::+:l_0=l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1::+: \text{nil}=l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1::+: (x::l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1::+:x:) L_E^{(l_1::+:x)::+:l_2=l_0} (x::l) L_{S1}^{\text{Rev}(l_1::+:x:, x::l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_S]]_p^- := \langle y_L :+ : x :, x :: y \rangle$$

$$[[L_B]]_u^- := y \quad [[L_S]]_{R_1}^- := \langle x, y_L, y \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1 :+ : x :) L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_S]]^+ := \lambda_y \langle y_{\perp} :+ : x :, x :: y_{\perp} \rangle$$

$$[[L_B]]_U^- := y_{\perp} \quad [[L_S]]_{R_1}^- := \langle x, y_{\perp}, y_{\perp} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil} M_E^{\text{nil} :+ : l_0 = l_0} \text{nil} R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} l_1 l_0 u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1 :+ : x :) L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$[[L_S]]^+ := \lambda_{x,l_2} \lambda_y \langle y_{\perp} :+ : x :, x :: y_{\perp} \rangle$$

$$[[L_B]]_u^- := y_{\perp} \quad [[L_S]]_{R_1}^- := \langle y_{\perp}, y_{\perp\perp\perp}, y_{\perp\perp\perp} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L l_0 \text{nil } M_E^{\text{nil} :+ : l_0 = l_0} \text{nil } R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} l_2 L_B L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat } l_1 l_0 u$$

$$L_S := \lambda_{x,l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p(l_1 :+ : x :) L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 x l_1 l w$$

## Extraction by Dialectica

Extracted program:

$$\begin{aligned} \llbracket L_S \rrbracket^+ &:= \lambda_{x,l_2} \lambda_y \langle y_{L:+:x:}, x :: y_{\downarrow} \rangle \\ \llbracket L_B \rrbracket_u^- &:= y_{\downarrow} \quad \llbracket L_S \rrbracket_{R_1}^- := \langle y_L, y_{\downarrow\downarrow L}, y_{\downarrow\downarrow\downarrow} \rangle \end{aligned}$$

$$\begin{aligned} \llbracket L \rrbracket_i^- &:= \mathcal{R}_{L(\rho)} I(\lambda_{y_A} \llbracket L_B \rrbracket_i^-) \\ &\left( \begin{array}{c} \llbracket L_S \rrbracket_i^- [y_S := \langle x, l, y \rangle] \\ \lambda_{x,l,p,y} \quad \begin{array}{c} i \\ \boxtimes \\ \rho(\llbracket L_S \rrbracket^+ x / y) \end{array} \end{array} \right) \end{aligned}$$

## Extraction by Dialectica

Extracted program:

$$\begin{aligned} \llbracket L_S \rrbracket^+ &:= \lambda_{x,l} \lambda_y \langle y_L := +:x:, x :: y \rangle \\ \llbracket L_B \rrbracket_u^- &:= y \quad \llbracket L_S \rrbracket_{R_1}^- := \langle y_L, y_{LL}, y_{LLL} \rangle \end{aligned}$$

$$\begin{aligned} \llbracket L \rrbracket_i^- &:= \mathcal{R}_{L(\rho)} I(\lambda_{y_A} \llbracket L_B \rrbracket_i^-) \\ &\left( \begin{array}{c} \llbracket L_S \rrbracket_i^- [y_S := \langle x, l, y \rangle] \\ \lambda_{x,l,p,y} \quad \begin{array}{c} i \\ \boxtimes \\ \rho(\llbracket L_S \rrbracket^+ x / y) \end{array} \end{array} \right) \end{aligned}$$

Cannot contract neither on  $u$ , nor on  $R_1$ !

## Extraction by Dialectica

Extracted program:

$$\llbracket L \rrbracket_u^- := \mathcal{R} \, l_2 \, (\lambda_y y_\perp) \, (\lambda_{x, l_2, \rho, y} \rho \langle y_{\perp} :+ : x :, x :: y_\perp \rangle)$$

$$\llbracket L_B \rrbracket_u^- := y_\perp \quad \llbracket L_S \rrbracket_{R_1}^- := \langle y_{\perp}, y_{\perp\perp}, y_{\perp\perp\perp} \rangle$$

$$\llbracket L \rrbracket_i^- := \mathcal{R}_{\perp(\rho)} \, l \, (\lambda_{y_A} \llbracket L_B \rrbracket_i^-) \left( \begin{array}{c} \llbracket L_S \rrbracket_i^- [y_S := \langle x, l, y \rangle] \\ \lambda_{x, l, \rho, y} \quad \begin{array}{c} i \\ \boxtimes \\ \rho(\llbracket L_S \rrbracket^+ x / y) \end{array} \end{array} \right)$$

Cannot contract neither on  $u$ , nor on  $R_1$ !

For  $u$  we don't need to contract

## Extraction by Dialectica

Extracted program:

$$R_1 : \quad \forall_{x,l_1,l_2}^U (\text{Rev}(l_1, l_2) \rightarrow \text{Rev}(l_1::x::, x::l_2))$$

$$\llbracket L_B \rrbracket_u^- := y \downarrow \quad \llbracket L_S \rrbracket_{R_1}^- := \langle y \downarrow, y \downarrow \downarrow, y \downarrow \downarrow \downarrow \rangle$$

$$\llbracket L \rrbracket_i^- := \mathcal{R}_{\perp(\rho)} / (\lambda_{y_A} \llbracket L_B \rrbracket_i^-) \left( \begin{array}{c} \llbracket L_S \rrbracket_i^- [y_S := \langle x, l, y \rangle] \\ \lambda_{x,l,p,y} \quad \quad \quad \begin{array}{c} i \\ \boxtimes \\ \rho(\llbracket L_S \rrbracket^+ x / y) \end{array} \end{array} \right)$$

Cannot contract neither on  $u$ , nor on  $R_1$ !

For  $u$  we don't need to contract, for  $R_1$  we need uniform quantifiers

## Extraction by Dialectica

Extracted program:

$$R_1 : \quad \forall_{x,l_1,l_2}^U (\text{Rev}(l_1, l_2) \rightarrow \text{Rev}(l_1::x::, x::l_2))$$

$$\llbracket L_B \rrbracket_u^- := y \downarrow \quad \llbracket L_S \rrbracket_{R_1}^- := \langle y \downarrow, y \downarrow \downarrow, y \downarrow \downarrow \downarrow \rangle$$

$$\llbracket L \rrbracket_i^- := \mathcal{R}_{\perp(\rho)} / (\lambda_{y_A} \llbracket L_B \rrbracket_i^-) \left( \begin{array}{c} \llbracket L_S \rrbracket_i^- [y_S := \langle x, l, y \rangle] \\ \lambda_{x,l,p,y} \quad \quad \quad \begin{array}{c} i \\ \boxtimes \\ \rho(\llbracket L_S \rrbracket^+ x / y) \end{array} \end{array} \right)$$

Cannot contract neither on  $u$ , nor on  $R_1$ !

For  $u$  we don't need to contract, for  $R_1$  we need uniform quantifiers

## Extraction by Dialectica

Extracted program:

$$[[L]]_u^- := \mathcal{R} \, l_2 \, (\lambda y y \perp) \, (\lambda x, l_2, p, y p \langle y \_ : + : x \_ , x \_ :: y \_ \rangle) \, y$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u \lambda_v^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} : + : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 : + : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 : + : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p \, (l_1 : + : x) \, L_E^{(l_1 : + : x) : + : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 : + : x, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l_w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket_u^- := \mathcal{R} \, l_0 \, (\lambda_y y \downarrow) \, (\lambda_{x, l_2, p, y} p \langle y \downarrow :+ : x :, x :: y \downarrow \rangle) \, y$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p \, (l_1 :+ : x :) \, L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket_u^- := \mathcal{R} \, l_0 \, (\lambda y y \perp) \, (\lambda_{x, l_2, p, y} p \langle y \perp :+ : x :, x :: y \perp \rangle) \, \langle \text{nil}, y \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} \, L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p \, (l_1 :+ : x :) \, L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket_u^- := \mathcal{R} \, l_0 \, (\lambda y y \perp) \, (\lambda_{x, l_2, p, y} p \langle y \perp :+ : x :, x :: y \perp \rangle) \, \langle \text{nil}, y \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u \lambda_v^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p \, (l_1 :+ : x :) \, L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket_u^- := \mathcal{R} \, l_0 \, (\lambda y y_\perp) \, (\lambda_{x, l_2, p, y} p \langle y_{\perp} :+ : x : , x :: y_\perp \rangle) \langle \text{nil}, \text{nil} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p \, (l_1 :+ : x : ) \, L_E^{(l_1 :+ : x : ) :+ : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 :+ : x : , x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l_w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket_u^- := \mathcal{R} \, l_0 \, (\lambda y y_\perp) \, (\lambda_{x, l_2, p, y} p \langle y_{\perp} :+ : x :, x :: y_\perp \rangle) \langle \text{nil}, \text{nil} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p(l_1 :+ : x :) L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l_w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket^+ := \mathcal{R} \, l_0 \, (\lambda y y \downarrow) \, (\lambda_{x, l_2, p, y} p \langle y \downarrow :+ : x :, x :: y \downarrow \rangle) \langle \text{nil}, \text{nil} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p \, (l_1 :+ : x :) \, L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l_w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket^+ := \lambda_{l_0} \mathcal{R} \, l_0 \, (\lambda_{y \perp} (\lambda_{x, l_2, p, y} p \langle y \perp :+ : x :, x :: y \perp \rangle)) \langle \text{nil}, \text{nil} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall_l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p(l_1 :+ : x :) L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket^+ := \lambda_{l_0} \mathcal{R} \, l_0 \, (\lambda_{y \downarrow} (\lambda_{x, l_2, p, y} p \langle y \downarrow :+ : x :, x :: y \downarrow \rangle)) \langle \text{nil}, \text{nil} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p(l_1 :+ : x :) L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} (x :: l) L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket^+ := \lambda_{l_0} \mathcal{R} \, l_0 \, (\lambda_y y_{\perp}) \, (\lambda_{x, l_2, p, y} p \langle y_{\perp} :+ : x :, x :: y_{\perp} \rangle) \langle \text{nil}, \text{nil} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u \forall_l (\text{Rev}(l_0, l) \rightarrow F) \, L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p(l_1 :+ : x :) \, L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket^+ := \lambda_{l_0} \mathcal{R} \, l_0 \, (\lambda_y \mathbf{y}_{\perp}) \, (\lambda_{x, l_2, p, y} p \langle \mathbf{y}_{\perp} \text{L} : + : x :, x :: \mathbf{y}_{\perp} \rangle) \, \langle \text{nil}, \text{nil} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall_l (\text{Rev}(l_0, l) \rightarrow F)} \, L \, l_0 \, \text{nil} \, M_E^{\text{nil} : + : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1} \lambda_v^{l_1 : + : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1} \lambda_v^{l_1 : + : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p(l_1 : + : x :) \, L_E^{(l_1 : + : x :) : + : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 : + : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

# Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket^+ := \lambda_{l_0} \mathcal{R} \, l_0 \, (\lambda_y y \perp) \, (\lambda_{x, l_2, p, y} p \langle y \perp :+ : x :, x :: y \perp \rangle) \langle \text{nil}, \text{nil} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} :+ : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1}^U \lambda_v^{l_1 :+ : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1}^U \lambda_v^{l_1 :+ : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)} \\ p \, (l_1 :+ : x :) \, L_E^{(l_1 :+ : x :) :+ : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 :+ : x :, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket^+ := \lambda_{l_0} \mathcal{R} \, l_0 \, (\lambda_y \mathbf{y} \perp) \, (\lambda_{x, l_2, p, y} p \langle \mathbf{y} \perp \vdash x, x \vdash \mathbf{y} \perp \rangle) \, \langle \mathbf{nil}, \mathbf{nil} \rangle$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \mathbf{nil} \, M_E^{\mathbf{nil} \vdash l_0 = l_0} \, \mathbf{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1}^U \lambda_v^{l_1 \vdash \mathbf{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1}^U \lambda_v^{l_1 \vdash (x \vdash l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p \, (l_1 \vdash x) \, L_E^{(l_1 \vdash x) \vdash l_2 = l_0} \, (x \vdash l) \, L_{S1}^{\text{Rev}(l_1 \vdash x, x \vdash l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket^+ := \lambda_{l_0} \mathcal{R} \, l_0 \, (\lambda_y y) \, (\lambda_{x, l_2, p, y} p(x :: y)) \, \text{nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil} :: + : l_0 = l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1}^U \lambda_v^{l_1 :: + : \text{nil} = l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1}^U \lambda_v^{l_1 :: + : (x :: l_2) = l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p \, (l_1 :: + : x ::) \, L_E^{(l_1 :: + : x ::) :: + : l_2 = l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1 :: + : x ::, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Extraction by Dialectica

Extracted program:

$$\llbracket M \rrbracket^+ := \lambda_{l_0} \mathcal{R} \, l_0 \, (\lambda_y y) \, (\lambda_{x, l_2, p, y} p(x :: y)) \, \text{nil}$$

$$\llbracket M \rrbracket^\circ := \lambda_{l_0} \mathcal{R} \, l_0 \, (\lambda_l l) \, (\lambda_{x, l_2, x_p, l} x_p(x :: l)) \, \text{nil}$$

$$M := \lambda_{R_0, R_1} \lambda_{l_0} \lambda_u^{\forall_l (\text{Rev}(l_0, l) \rightarrow F)} L \, l_0 \, \text{nil} \, M_E^{\text{nil}:+:l_0=l_0} \, \text{nil} \, R_0$$

$$L := \lambda_{l_2} \text{Ind}_{L(N)} \, l_2 \, L_B \, L_S$$

$$L_B := \lambda_{l_1}^U \lambda_v^{l_1:+:\text{nil}=l_0} \text{Compat} \, l_1 \, l_0 \, u$$

$$L_S := \lambda_{x, l_2} \lambda_p \lambda_{l_1}^U \lambda_v^{l_1:+:(x :: l_2)=l_0} \lambda_l \lambda_w^{\text{Rev}(l_1, l)}$$

$$p \, (l_1:+:x:) \, L_E^{(l_1:+:x):+:l_2=l_0} \, (x :: l) \, L_{S1}^{\text{Rev}(l_1:+:x:, x :: l)}$$

$$L_{S1} := R_1 \, x \, l_1 \, l \, w$$

## Conclusions and future work

### Uniformities

- ▶ **are quantifier annotations**
- ▶ express lack of computational dependence
- ▶ can remove unnecessary parameters
- ▶ can improve complexity
- ▶ can remove unnecessary contractions in Dialectica
- ▶ could be refined for Dialectica
- ▶ could be inferred automatically for a given proof of a fixed formula

## Conclusions and future work

### Uniformities

- ▶ are quantifier annotations
- ▶ express lack of computational dependence
- ▶ can remove unnecessary parameters
- ▶ can improve complexity
- ▶ can remove unnecessary contractions in Dialectica
- ▶ could be refined for Dialectica
- ▶ could be inferred automatically for a given proof of a fixed formula

## Conclusions and future work

### Uniformities

- ▶ are quantifier annotations
- ▶ express lack of computational dependence
- ▶ can remove unnecessary parameters
- ▶ can improve complexity
- ▶ can remove unnecessary contractions in Dialectica
- ▶ could be refined for Dialectica
- ▶ could be inferred automatically for a given proof of a fixed formula

## Conclusions and future work

### Uniformities

- ▶ are quantifier annotations
- ▶ express lack of computational dependence
- ▶ can remove unnecessary parameters
- ▶ can improve complexity
- ▶ can remove unnecessary contractions in Dialectica
- ▶ could be refined for Dialectica
- ▶ could be inferred automatically for a given proof of a fixed formula

## Conclusions and future work

### Uniformities

- ▶ are quantifier annotations
- ▶ express lack of computational dependence
- ▶ can remove unnecessary parameters
- ▶ can improve complexity
- ▶ can remove unnecessary contractions in Dialectica
- ▶ could be refined for Dialectica
- ▶ could be inferred automatically for a given proof of a fixed formula

## Conclusions and future work

### Uniformities

- ▶ are quantifier annotations
- ▶ express lack of computational dependence
- ▶ can remove unnecessary parameters
- ▶ can improve complexity
- ▶ can remove unnecessary contractions in Dialectica
- ▶ could be refined for Dialectica
- ▶ could be inferred automatically for a given proof of a fixed formula

## Conclusions and future work

### Uniformities

- ▶ are quantifier annotations
- ▶ express lack of computational dependence
- ▶ can remove unnecessary parameters
- ▶ can improve complexity
- ▶ can remove unnecessary contractions in Dialectica
- ▶ could be refined for Dialectica
- ▶ could be inferred automatically for a given proof of a fixed formula

Thank you

Thank you for your attention