Deduction and Presentation in ρ*Log*

Mircea Marin
Johann Radon Institute for Computational and Applied Mathematics
Austrian Academy of Sciences
Linz, Austria

Florina Piroi
Research Institute for Symbolic Computation
Johannes Kepler University
Hagenberg, Austria

Outline

- What is ρ*Log*?
- Rule–based programming in ρ*Log*
- Deductive and proof capabilities
- Illustrative examples
- Conclusion and future work

What is *ρLog*?

- FunLog renamed to *ρLog*
- a system for rule–based programming, implemented in *Mathematica*.
- rule: labeled specification of a partially defined,

  possibly non–deterministic computation:

$$\underbrace{\mathtt{l}}_{\text{name}} :: \underbrace{\mathtt{patt} \longrightarrow \mathtt{rhs}}_{\text{code}}$$

patt – specifies the shape of an expression

rhs – expression encoding a computation

Rule-based programming in *ρLog*

Basic Usage

Rules are applied to input expressions

```
ApplyRule[expr, l];
ApplyRuleList[expr, l];
```

Example: a non–deteministic rule

select an arbitrary argument t_i of an expression of the form $\mathrm{f}(t_1, \ldots, t_n)$, f constant

$\mathtt{select} :: \mathtt{f}(\ldots, x, \ldots) \longrightarrow x$

$\mathtt{f}(\ldots, x, \ldots) \longrightarrow_{\mathtt{select}} x$

$\mathtt{f}(2, 5, a, 10) \longrightarrow_{\mathtt{select}} 2$

```
<< RhoLog`RhoLog`(* loading the system *)
```

```
DeclareRule[f[___, x_, ___] :→ x, "select"];
```

```
E1 = f[2, 5, a, 10, 2 + b];
```

```
ApplyRule[E1, "select"]
```

```
– found–
2
```

```
ApplyRuleList[E1, "select"]
```

```
{2, 5, a, 10, 2 + b}
```

Example: a partial defined rule

select an arbitrary argument t_i of an expression of the form $f(t_1, \ldots, t_n)$, f a constant

select :: f (…, *x*, …) ⟶ *x*

```
E2 = g[a, b, c];
```

```
ApplyRule[E2, "select"]
```

```
– not found–
g[a, b, c]
```

Remark

The binary reduction relation $\longrightarrow_l$ associated with a rule l can be:

- partially defined and
- non–deterministic.

Rule-based programming in ρLog (ctd.)

Combining rules

Alternatives: $\longrightarrow_{l_1 | \ldots | l_n} := \longrightarrow_{l_1} \bigcup \ldots \bigcup \longrightarrow_{l_n}$

Composition: $\longrightarrow_{l_1 \circ l_2} := \longrightarrow_{l_1} \circ \longrightarrow_{l_2}$

Using reflexive–transitive closures:

$$\longrightarrow_{\mathbf{Repeat}[l_1, l_2]} := \longrightarrow_{l_1}^{*} \circ \longrightarrow_{l_2},$$

$$\longrightarrow_{\mathbf{Until}[l_2, l_1]} := \longrightarrow_{l_1}^{*} \circ \longrightarrow_{l_2}$$

Normal forms: $\longrightarrow_{\mathbf{NFQ}[l]} := \{(E, E) \mid \nexists E' : E \longrightarrow_{l} E'\},$

$$\longrightarrow_{\mathbf{NF}[l]} := \longrightarrow_{l}^{*} \circ \longrightarrow_{\mathbf{NFQ}[l]}$$

Rewrite rule induced by l:

$$\longrightarrow_{\mathbf{Rw}[l]} := \{(E, E'') \mid \exists\, p : (E|_{p} \longrightarrow_{l} E') \bigwedge (E'' = E[E']_{p})\}$$

in ρLog, Rw[l] is the l_1 declared by RwRule[l, l_1]

Aliases: $\longrightarrow_{l} := \longrightarrow_{l_1}$ if l is an alias of l_1

Programming Example 1

Sorting integers by means of sequence variables

$$\{\overline{x}, m, \overline{y}, n, \overline{z}\} \longrightarrow_{\text{b-sort}} \{\overline{x}, n, \overline{y}, m, \overline{z}\} \text{ if } m > n$$

```
DeclareRule[{x___, m_, y___, n_, z___} /; (m > n) :→ {x, n, y, m, z}, "b-sort"];
L := {30, 4, 13, 26, 8, 45};
```

1st Variant

Repeat[b–sort, Identity]

```
DeclareRule[x_ :→ x, "Identity"];
ApplyRule[L, Repeat["b-sort", "Identity"]]
```

{4, 8, 13, 26, 30, 45}

2nd Variant

L $\longrightarrow_{\mathbf{NF[b\text{–}sort]}}$ L' iff L' is the sorted version of L; $\longrightarrow_{\mathbf{NF[b\text{–}sort]}}$ is confluent

```
ApplyRule[L, NF["b-sort"]]
```

{4, 8, 13, 26, 30, 45}

Programming Example 2

Joinability test in non–commutative group theory

$$f(f(x,y),z) \longrightarrow_A f(x, f(y,z)),$$
$$f(x,e) \longrightarrow_N x,$$
$$f(x,i(x)) \longrightarrow_I e.$$

```
DeclareRule[f[f[x_, y_], z_] :> f[x, f[y, z]], "A"];
DeclareRule[f[x_, e] :> x, "N"];
DeclareRule[f[x_, i[x_]] :> e, "I"];
```

Decide whether s = f(f(a,e),i(a)) and t = f(f(e,b),i(b)) are joinable in the group theory.

```
SetAlias["A" | "N" | "I", "G"];
RWRule["G", "Group", Traversal → "LeftOut"];
DeclareRule[eq[x_, x_] :> True, "Eq"];
SetAlias[Until["Eq", "Group"], "Join"];
```

```
s := f[f[a, e], i[a]];
t := f[f[e, b], i[b]];
P1 := eq[s, t];
```

The following call will return True iff s and t are joinable.

```
ApplyRule[P1, "Join"]
```

```
--found--
```

```
True
```

Why? Because $\longrightarrow_{\text{Join}} = \longrightarrow^*_{\text{Group}} \circ \longrightarrow_{\text{Eq}}$ and

eq(f(**f(a,e)**,i(x)),f(f(e,y),i(y)))$\longrightarrow_{\text{Group}}$

eq(**f(a,i(a))**,f(f(e,y),i(y)))$\longrightarrow_{\text{Group}}$

eq(e,**f(f(e,b),i(b))**)$\longrightarrow_{\text{Group}}$

eq(e,f(e,**f(b,i(b))**))$\longrightarrow_{\text{Group}}$

eq(e,**f(e,e)**)$\longrightarrow_{\text{Group}}$

eq(e,e)$\longrightarrow_{\text{Eq}}$True

Deductive and Proof Capabilities

Given an expression E and a rule $\boldsymbol{l}$

Decide whether the formula $\exists x : \mathrm{E} \longrightarrow_{\boldsymbol{l}} x$ is valid or not.

If requested, provide the following:

(1) a witness E' for x, such that $\mathrm{E} \longrightarrow_{\boldsymbol{l}} \mathrm{E}'$,

(2) the list $\{\mathrm{E}' \mid \mathrm{E} \longrightarrow_{\boldsymbol{l}} \mathrm{E}'\}$,

(3) a proof for the decision.

- 6 inference rules in ρ*Log*

Deductive and Proof Capabilities (1)

Given an expression E and a rule ι

Decide whether the formula $\exists x : E \longrightarrow_{\iota} x$ is valid or not.

If requested, provide the following:

(1) a witness E' for x, such that $E \longrightarrow_{\iota} E'$,

```
ApplyRule[E, l]
```

gives (1) E' s.t. $E \longrightarrow_{l} E'$ if $\exists x : E \longrightarrow_{l} x$ is valid,

(2) E, otherwise.

$x \wedge y \longrightarrow_{C} y \wedge x$

```
ApplyRule[ A ∧ B, NF[ "c"] ];
```

```
ApplyRule[E, l, MaxDepth ⟶ value]
```

(2) the list $\{E' \mid E \longrightarrow_{\iota} E'\}$,

(3) a proof for the decision.

Deductive and Proof Capabilities (2)

Given an expression E and a rule l

Decide whether the formula $\exists x : E \longrightarrow_l x$ is valid or not.

If requested, provide the following:

(1) a witness E' for x, such that $E \longrightarrow_l E'$,

(2) the list $\{E' \mid E \longrightarrow_l E'\}$,

ApplyRuleList[E, *l*]

gives the (possibly empty) list $\{E' \mid E \longrightarrow_l E'\}$.

```
ApplyRuleList[eq[f[f[e, y], i[y]], x], "Join"]
```

ApplyRuleList[E, *l*, MaxSols ⟶ *nnn*, MaxDepth ⟶ *value*]

gives the first *nnn* values of $\{E' \mid E \longrightarrow_l E'\}$.

```
Print[E1];
Print[ApplyRuleList[E1, "select"]];
ApplyRuleList[E1, "select", MaxSols→2]
```

- $\{E' \mid E \longrightarrow_l E'\} \neq \{\}$ then $\nexists x : E \longrightarrow_l x$ is valid.
- The search is exhaustive

(3) a proof for the decision.

Deductive and Proof Capabilities (3)

Given an expression E and a rule ι

Decide whether the formula $\exists x : E \longrightarrow_{\iota} x$ is valid or not.

If requested, provide the following:

(1) a witness E' for x, such that $E \longrightarrow_{\iota} E'$,

(2) the list $\{E' \mid E \longrightarrow_{\iota} E'\}$,

(3) a proof for the decision.

ApplyRule[E, l , TraceStyle ⟶ *value*]
ApplyRuleList[E, l , TraceStyle ⟶ *value*]

value ∈ { "None","Object","Compact","Verbose"}

- a proof object generated
- proof objects are the internal encodings of deduction trees.

Proofs: Example

$$
\begin{array}{lcl}
A & :: & f(f(x, y), z) \longrightarrow f(x, f(y, z)) \\
N & :: & f(x, e) \longrightarrow x \\
I & :: & f(x, i(x)) \longrightarrow e \\
Eq & :: & =(x, x) \longrightarrow \text{True} \\
G & :: & A \mid N \mid I \\
Group & :: & \text{Rw}[G] \\
Join & :: & \text{Repeat}[Group, Eq]
\end{array}
$$

$$\exists\, x :\; =(f(f(a, e), i(a)), f(f(e, b), i(b))) \longrightarrow_{\text{Join}} x$$

```
P1
```

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]]
```

"Verbose" presentation

```
ApplyRule[P1, "Join", TraceStyle → "Verbose"]
```

– *success* –

See Appendix 1 for the content of the output notebook.

"Compact" presentation

```
ApplyRule[P1, "Join", TraceStyle → "Compact"]
```

– *success* –

See Appendix 2 for the content of the output notebook.

```
ApplyRule[eq[f[f[a, e], i[a]], c], "Join", TraceStyle → "Compact"]
```

– *failure* –

Deduction Tree and Proof Example

$$
\begin{array}{ll}
f_1 : (-\infty, 0) \to \mathbb{R}, & f_1(x) = x + 7 \\
f_2 : (-\infty, 1) \to \mathbb{R}, & f_2(x) = x + 4 \\
g : (0, \infty) \to \mathbb{R}, & g(x) = x/2
\end{array}
$$

$$
\begin{array}{l}
f : \mathbb{R} \to \mathbb{R},\ f(x) = f_1(x) \text{ or } f_2(x), \\
h : \mathbb{R} \to \mathbb{R},\ h(x) = g(f(x)).
\end{array}
$$

Problem: Program a rule that behaves like 'h'.

Answer: Rule "fg" declared below.

```
DeclareRule[x_Real /; (x < 0) :> x + 7, "f1"];
DeclareRule[x_Real /; (x < 1) :> x + 4, "f2"];
DeclareRule[x_Real /; (x > 0) :> x / 2, "g"];
SetAlias["f1" | "f2", "f"];
SetAlias["f" ∘ "g", "fg"];
```

The deduction tree for $\exists\, x : -4.2 \longrightarrow_{\text{fg}} x$ is:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(-4.2 \to_{\text{"f1"}} 2.8) \wedge \dfrac{2.8 \to_{\text{"g"}} 1.4}{\exists^? x : 2.8 \to_{\text{"g"}} x}}{\exists^? x : -4.2 \to_{\text{"f1"}\circ\text{"g"}} x} \quad \dfrac{(-4.2 \to_{\text{"f2"}} -0.2) \wedge \dfrac{}{\exists^? x : -0.2 \to_{\text{"g"}} x}}{\exists^? x : -4.2 \to_{\text{"f2"}\circ\text{"g"}} x}}{\exists^? x : -4.2 \to_{(\text{"f1"}\circ\text{"g"})|(\text{"f2"}\circ\text{"g"})} x}}{\exists^? x : -4.2 \to_{(\text{"f1"}|\text{"f2"})\circ\text{"g"}} x}}{\exists^? x : -4.2 \to_{\text{"f"}\circ\text{"g"}} x}}{\exists^? x : -4.2 \to_{\text{"fg"}} x}
$$

Presentation styles

"Compact"

```
ApplyRule[ 4.2, "fg", TraceStyle → "Compact"];
```

See appendix 3 for the content of the output notebook.

"Verbose"

```
ApplyRule[ 4.2, "fg", TraceStyle → "Verbose"];
```

See appendix 4 for the content of the output notebook.

"Object"

```
ApplyRule[ 4.2, "fg", TraceStyle → "Object"];
```

```
$SNODE({-4.2, {fg, f∘g, (f1 | f2)∘g, f1∘g | f2∘g}, 1.4},
  $SNODE({-4.2, f1∘g, 1.4}, $SNODE({-4.2, ⟨f1, 2.8, g⟩, 1.4})))
```

Conclusions and Future Work

- modelling non–deterministic and/or partial computations

- ρLog as a reasoner programming tool

- control mechanisms for pattern matching with sequence variables (Sequentica)

- http://heaven.ricam.uni-linz.ac.at/people/page/marin/RhoLog/

Appendix 1

Find an expression *e* such that:

eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] $\underset{\texttt{Join}}{\longrightarrow}$ *e*

Solution(s):

True

Justification:

eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] $\underset{\texttt{Join}}{\longrightarrow}$?

which is

eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] $\underset{\texttt{Eq|Group∘Until[Eq,Group]}}{\longrightarrow}$?

because

Join $\overset{\texttt{def}}{\longmapsto}$ Until[Eq, Group] $\overset{\texttt{def}}{\longmapsto}$ Eq | Group ∘ Until[Eq, Group]

Then

eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] $\underset{\texttt{Group∘Until[Eq,Group]}}{\longrightarrow}$?

which is

eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] $\underset{\texttt{Rw∘((G|SEL[Group])∘Until[Eq,Group])}}{\longrightarrow}$?

because

Group ∘ Until[Eq, Group] $\overset{\texttt{def}}{\longmapsto}$ (Rw ∘ (**G** | SEL[Group])) ∘ Until[Eq, Group] $\overset{\texttt{def}}{\longmapsto}$
Rw ∘ ((**G** | SEL[Group]) ∘ Until[Eq, Group])

Then

eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] $\underset{\texttt{(G|SEL[Group])∘Until[Eq,Group]}}{\longrightarrow}$?

which is

eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] $\underset{\texttt{G∘Until[Eq,Group]|SEL[Group]∘Until[Eq,Group]}}{\longrightarrow}$?

because

(**G** | SEL[Group]) ∘ Until[Eq, Group] $\overset{\texttt{def}}{\longmapsto}$
G ∘ Until[Eq, Group] | SEL[Group] ∘ Until[Eq, Group]

Then

eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] $\underset{\texttt{SEL[Group]∘Until[Eq,Group]}}{\longrightarrow}$?

We choose a subexpression:

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] ⟶SEL[Group]
  eq[f[f[a, e], i[a]], f[f[e, b], i[b]]]
```

and

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] ⟶(G|SEL[Group])∘Until[Eq,Group] ?
```

which is

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] ⟶G∘Until[Eq,Group]|SEL[Group]∘Until[Eq,Group] ?
```

because

```
(G | SEL[Group]) ∘ Until[Eq, Group] ⟼def
  G ∘ Until[Eq, Group] | SEL[Group] ∘ Until[Eq, Group]
```

Then

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] ⟶SEL[Group]∘Until[Eq,Group] ?
```

We choose a subexpression:

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] ⟶SEL[Group]
  eq[f[f[a, e], i[a]], f[f[e, b], i[b]]]
```

and

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] ⟶(G|SEL[Group])∘Until[Eq,Group] ?
```

which is

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] ⟶G∘Until[Eq,Group]|SEL[Group]∘Until[Eq,Group] ?
```

because

```
(G | SEL[Group]) ∘ Until[Eq, Group] ⟼def
  G ∘ Until[Eq, Group] | SEL[Group] ∘ Until[Eq, Group]
```

Then

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] ⟶G∘Until[Eq,Group] ?
```

which is

```
eq[f[f[a, e], i[a]],
  f[f[e, b], i[b]]] ⟶A∘Until[Eq,Group]|N∘Until[Eq,Group]|I∘Until[Eq,Group] ?
```

because

```
G ∘ Until[Eq, Group] ⟼def
  (A | N | I) ∘ Until[Eq, Group] ⟼def (A | N | I) ∘ Until[Eq, Group] ⟼def
  A ∘ Until[Eq, Group] | N ∘ Until[Eq, Group] | I ∘ Until[Eq, Group]
```

Then

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] --N∘Until[Eq,Group]--> ?
```

We can rewrite:

```
eq[f[f[a, e], i[a]], f[f[e, b], i[b]]] --N--> eq[f[a, i[a]], f[f[e, b], i[b]]]
```

and

```
eq[f[a, i[a]], f[f[e, b], i[b]]] --Until[Eq,Group]--> ?
```

which is

```
eq[f[a, i[a]], f[f[e, b], i[b]]] --Eq|Group∘Until[Eq,Group]--> ?
```

because

```
Until[Eq, Group] ⟼def Eq | Group∘Until[Eq, Group]
```

Then

```
eq[f[a, i[a]], f[f[e, b], i[b]]] --Group∘Until[Eq,Group]--> ?
```

which is

```
eq[f[a, i[a]], f[f[e, b], i[b]]] --Rw∘((G|SEL[Group])∘Until[Eq,Group])--> ?
```

because

```
Group∘Until[Eq, Group] ⟼def (Rw∘(G | SEL[Group]))∘Until[Eq, Group] ⟼def
 Rw∘((G | SEL[Group])∘Until[Eq, Group])
```

Then

```
eq[f[a, i[a]], f[f[e, b], i[b]]] --(G|SEL[Group])∘Until[Eq,Group]--> ?
```

which is

```
eq[f[a, i[a]], f[f[e, b], i[b]]] --G∘Until[Eq,Group]|SEL[Group]∘Until[Eq,Group]--> ?
```

because

```
(G | SEL[Group])∘Until[Eq, Group] ⟼def
 G∘Until[Eq, Group] | SEL[Group]∘Until[Eq, Group]
```

Then

```
eq[f[a, i[a]], f[f[e, b], i[b]]] --SEL[Group]∘Until[Eq,Group]--> ?
```

We choose a subexpression:

```
eq[f[a, i[a]], f[f[e, b], i[b]]] --SEL[Group]--> eq[f[a, i[a]], f[f[e, b], i[b]]]
```

and

```
eq[f[a, i[a]], f[f[e, b], i[b]]] --(G|SEL[Group])∘Until[Eq,Group]--> ?
```

which is

eq[**f[a, i[a]]**, f[f[e, b], i[b]]] $\xrightarrow[\texttt{G}\circ\texttt{Until[Eq,Group]|SEL[Group]}\circ\texttt{Until[Eq,Group]}]{}$?

because

(**G** | SEL[Group]) ∘ Until[Eq, Group] $\xmapsto{\texttt{def}}$
G ∘ Until[Eq, Group] | SEL[Group] ∘ Until[Eq, Group]

Then

eq[**f[a, i[a]]**, f[f[e, b], i[b]]] $\xrightarrow[\texttt{G}\circ\texttt{Until[Eq,Group]}]{}$?

which is

eq[**f[a, i[a]]**, f[f[e, b], i[b]]] $\xrightarrow[\texttt{A}\circ\texttt{Until[Eq,Group]|N}\circ\texttt{Until[Eq,Group]|I}\circ\texttt{Until[Eq,Group]}]{}$?

because

G ∘ Until[Eq, Group] $\xmapsto{\texttt{def}}$
(**A | N | I**) ∘ Until[Eq, Group] $\xmapsto{\texttt{def}}$ (**A** | **N** | **I**) ∘ Until[Eq, Group] $\xmapsto{\texttt{def}}$
A ∘ Until[Eq, Group] | **N** ∘ Until[Eq, Group] | **I** ∘ Until[Eq, Group]

Then

eq[**f[a, i[a]]**, f[f[e, b], i[b]]] $\xrightarrow[\texttt{I}\circ\texttt{Until[Eq,Group]}]{}$?

We can rewrite:

eq[**f[a, i[a]]**, f[f[e, b], i[b]]] $\xrightarrow[\texttt{I}]{}$ eq[e, f[f[e, b], i[b]]]

and

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\texttt{Until[Eq,Group]}]{}$?

which is

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\texttt{Eq|Group}\circ\texttt{Until[Eq,Group]}]{}$?

because

Until[Eq, Group] $\xmapsto{\texttt{def}}$ Eq | Group ∘ Until[Eq, Group]

Then

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\texttt{Group}\circ\texttt{Until[Eq,Group]}]{}$?

which is

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\texttt{Rw}\circ\texttt{((G|SEL[Group])}\circ\texttt{Until[Eq,Group])}]{}$?

because

Group ∘ Until[Eq, Group] $\xmapsto{\texttt{def}}$ (Rw ∘ (**G** | SEL[Group])) ∘ Until[Eq, Group] $\xmapsto{\texttt{def}}$
Rw ∘ ((**G** | SEL[Group]) ∘ Until[Eq, Group])

Then

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{(G|SEL[Group])∘Until[Eq,Group]}]{}$?

which is

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{G∘Until[Eq,Group]|SEL[Group]∘Until[Eq,Group]}]{}$?

because

(G | SEL[Group]) ∘ Until[Eq, Group] $\xmapsto{\text{def}}$
G ∘ Until[Eq, Group] | SEL[Group] ∘ Until[Eq, Group]

Then

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{SEL[Group]∘Until[Eq,Group]}]{}$?

We choose a subexpression:

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{SEL[Group]}]{}$ eq[e, f[f[e, b], i[b]]]

and

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{(G|SEL[Group])∘Until[Eq,Group]}]{}$?

which is

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{G∘Until[Eq,Group]|SEL[Group]∘Until[Eq,Group]}]{}$?

because

(G | SEL[Group]) ∘ Until[Eq, Group] $\xmapsto{\text{def}}$
G ∘ Until[Eq, Group] | SEL[Group] ∘ Until[Eq, Group]

Then

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{G∘Until[Eq,Group]}]{}$?

which is

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{A∘Until[Eq,Group]|N∘Until[Eq,Group]|I∘Until[Eq,Group]}]{}$?

because

G ∘ Until[Eq, Group] $\xmapsto{\text{def}}$
(A | N | I) ∘ Until[Eq, Group] $\xmapsto{\text{def}}$ (A | N | I) ∘ Until[Eq, Group] $\xmapsto{\text{def}}$
A ∘ Until[Eq, Group] | N ∘ Until[Eq, Group] | I ∘ Until[Eq, Group]

Then

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{A∘Until[Eq,Group]}]{}$?

We can rewrite:

eq[e, f[f[e, b], i[b]]] $\xrightarrow[\text{A}]{}$ eq[e, f[e, f[b, i[b]]]] and

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{Until[Eq,Group]}]{} ?$$

which is

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{Eq|Group}\circ\texttt{Until[Eq,Group]}]{} ?$$

because

$$\texttt{Until[Eq, Group]} \xmapsto{\texttt{def}} \texttt{Eq | Group} \circ \texttt{Until[Eq, Group]}$$

Then

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{Group}\circ\texttt{Until[Eq,Group]}]{} ?$$

which is

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{Rw}\circ\texttt{((G|SEL[Group])}\circ\texttt{Until[Eq,Group])}]{} ?$$

because

$$\texttt{Group} \circ \texttt{Until[Eq, Group]} \xmapsto{\texttt{def}} \texttt{(Rw} \circ \texttt{(G | SEL[Group]))} \circ \texttt{Until[Eq, Group]} \xmapsto{\texttt{def}} \texttt{Rw} \circ \texttt{((G | SEL[Group])} \circ \texttt{Until[Eq, Group])}$$

Then

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{(G|SEL[Group])}\circ\texttt{Until[Eq,Group]}]{} ?$$

which is

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{G}\circ\texttt{Until[Eq,Group]|SEL[Group]}\circ\texttt{Until[Eq,Group]}]{} ?$$

because

$$\texttt{(G | SEL[Group])} \circ \texttt{Until[Eq, Group]} \xmapsto{\texttt{def}} \texttt{G} \circ \texttt{Until[Eq, Group] | SEL[Group]} \circ \texttt{Until[Eq, Group]}$$

Then

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{SEL[Group]}\circ\texttt{Until[Eq,Group]}]{} ?$$

We choose a subexpression:

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{SEL[Group]}]{} \texttt{eq[e, f[e, f[b, i[b]]]]}$$

and

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{(G|SEL[Group])}\circ\texttt{Until[Eq,Group]}]{} ?$$

which is

$$\texttt{eq[e, f[e, f[b, i[b]]]]} \xrightarrow[\texttt{G}\circ\texttt{Until[Eq,Group]|SEL[Group]}\circ\texttt{Until[Eq,Group]}]{} ?$$

because

(G | SEL[Group]) ∘ Until[Eq, Group] $\xmapsto{\text{def}}$
G ∘ Until[Eq, Group] | SEL[Group] ∘ Until[Eq, Group]

Then

eq[e, f[e, f[b, i[b]]]] $\xrightarrow[\text{SEL[Group]∘Until[Eq,Group]}]{}$?

We choose a subexpression:

eq[e, f[e, f[b, i[b]]]] $\xrightarrow[\text{SEL[Group]}]{}$ eq[e, f[e, f[b, i[b]]]]

and

eq[e, f[e, f[b, i[b]]]] $\xrightarrow[\text{(G|SEL[Group])∘Until[Eq,Group]}]{}$?

which is

eq[e, f[e, f[b, i[b]]]] $\xrightarrow[\text{G∘Until[Eq,Group]|SEL[Group]∘Until[Eq,Group]}]{}$?

because

(G | SEL[Group]) ∘ Until[Eq, Group] $\xmapsto{\text{def}}$
G ∘ Until[Eq, Group] | SEL[Group] ∘ Until[Eq, Group]

Then

eq[e, f[e, f[b, i[b]]]] $\xrightarrow[\text{G∘Until[Eq,Group]}]{}$?

which is

eq[e, f[e, f[b, i[b]]]] $\xrightarrow[\text{A∘Until[Eq,Group]|N∘Until[Eq,Group]|I∘Until[Eq,Group]}]{}$?

because

G ∘ Until[Eq, Group] $\xmapsto{\text{def}}$
(A | N | I) ∘ Until[Eq, Group] $\xmapsto{\text{def}}$ (A | N | I) ∘ Until[Eq, Group] $\xmapsto{\text{def}}$
A ∘ Until[Eq, Group] | N ∘ Until[Eq, Group] | I ∘ Until[Eq, Group]

Then

eq[e, f[e, f[b, i[b]]]] $\xrightarrow[\text{I∘Until[Eq,Group]}]{}$?

We can rewrite:

eq[e, f[e, f[b, i[b]]]] $\xrightarrow[\text{I}]{}$ eq[e, f[e, e]]

and

eq[e, f[e, e]] $\xrightarrow[\text{Until[Eq,Group]}]{}$?

which is

eq[e, f[e, e]] $\xrightarrow[\text{Eq|Group∘Until[Eq,Group]}]{}$?

because

$$\texttt{Until[Eq, Group]} \xmapsto{\text{def}} \texttt{Eq | Group} \circ \texttt{Until[Eq, Group]}$$

Then

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{Group}\circ\texttt{Until[Eq,Group]}}]{} \ ?$$

which is

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{Rw}\circ\texttt{((G|SEL[Group])}\circ\texttt{Until[Eq,Group])}}]{} \ ?$$

because

$$\texttt{Group}\circ\texttt{Until[Eq, Group]} \xmapsto{\text{def}} \texttt{(Rw}\circ\texttt{(G | SEL[Group]))}\circ\texttt{Until[Eq, Group]} \xmapsto{\text{def}} \texttt{Rw}\circ\texttt{((G | SEL[Group])}\circ\texttt{Until[Eq, Group])}$$

Then

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{(G|SEL[Group])}\circ\texttt{Until[Eq,Group]}}]{} \ ?$$

which is

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{G}\circ\texttt{Until[Eq,Group]|SEL[Group]}\circ\texttt{Until[Eq,Group]}}]{} \ ?$$

because

$$\texttt{(G | SEL[Group])}\circ\texttt{Until[Eq, Group]} \xmapsto{\text{def}} \texttt{G}\circ\texttt{Until[Eq, Group] | SEL[Group]}\circ\texttt{Until[Eq, Group]}$$

Then

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{SEL[Group]}\circ\texttt{Until[Eq,Group]}}]{} \ ?$$

We choose a subexpression:

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{SEL[Group]}}]{} \texttt{eq[e, f[e, e]]}$$

and

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{(G|SEL[Group])}\circ\texttt{Until[Eq,Group]}}]{} \ ?$$

which is

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{G}\circ\texttt{Until[Eq,Group]|SEL[Group]}\circ\texttt{Until[Eq,Group]}}]{} \ ?$$

because

$$\texttt{(G | SEL[Group])}\circ\texttt{Until[Eq, Group]} \xmapsto{\text{def}} \texttt{G}\circ\texttt{Until[Eq, Group] | SEL[Group]}\circ\texttt{Until[Eq, Group]}$$

Then

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{G}\circ\texttt{Until[Eq,Group]}}]{} \ ?$$

which is

$$\texttt{eq[e, f[e, e]]} \xrightarrow[{\texttt{A}\circ\texttt{Until[Eq,Group]|N}\circ\texttt{Until[Eq,Group]|I}\circ\texttt{Until[Eq,Group]}}]{} \ ?$$

because

$$\mathbf{G} \circ \mathtt{Until}[\mathtt{Eq}, \mathtt{Group}] \overset{\mathtt{def}}{\longmapsto} (\mathbf{A} \mid \mathbf{N} \mid \mathbf{I}) \circ \mathtt{Until}[\mathtt{Eq}, \mathtt{Group}] \overset{\mathtt{def}}{\longmapsto} (\mathbf{A} \mid \mathbf{N} \mid \mathbf{I}) \circ \mathtt{Until}[\mathtt{Eq}, \mathtt{Group}] \overset{\mathtt{def}}{\longmapsto} \mathbf{A} \circ \mathtt{Until}[\mathtt{Eq}, \mathtt{Group}] \mid \mathbf{N} \circ \mathtt{Until}[\mathtt{Eq}, \mathtt{Group}] \mid \mathbf{I} \circ \mathtt{Until}[\mathtt{Eq}, \mathtt{Group}]$$

Then

$$\mathtt{eq}[\mathtt{e}, \mathbf{f[e, e]}] \xrightarrow[\mathbf{N} \circ \mathtt{Until}[\mathtt{Eq},\mathtt{Group}]]{} ?$$

We can rewrite:

$$\mathtt{eq}[\mathtt{e}, \mathbf{f[e, e]}] \xrightarrow[\mathbf{N}]{} \mathtt{eq}[\mathtt{e}, \mathtt{e}]$$

and

$$\mathtt{eq}[\mathtt{e}, \mathtt{e}] \xrightarrow[\mathtt{Until}[\mathtt{Eq},\mathtt{Group}]]{} ?$$

which is

$$\mathtt{eq}[\mathtt{e}, \mathtt{e}] \xrightarrow[\mathtt{Eq} \mid \mathtt{Group} \circ \mathtt{Until}[\mathtt{Eq},\mathtt{Group}]]{} ?$$

because

$$\mathtt{Until}[\mathtt{Eq}, \mathtt{Group}] \overset{\mathtt{def}}{\longmapsto} \mathtt{Eq} \mid \mathtt{Group} \circ \mathtt{Until}[\mathtt{Eq}, \mathtt{Group}]$$

Finally we have

$$\mathtt{eq}[\mathtt{e}, \mathtt{e}] \xrightarrow[\mathtt{Eq}]{} \mathtt{True}$$

□

Appendix 2

Find an expression *e* such that:

$\texttt{eq[f[f[a, e], i[a]], f[f[e, b], i[b]]]} \underset{\texttt{Join}}{\longrightarrow} e$

Solution(s):

```
True
```

Justification:

$\texttt{eq[f[}\mathbf{\texttt{f[a, e]}}\texttt{, i[a]], f[f[e, b], i[b]]]} \underset{\mathbf{N}}{\longrightarrow} \texttt{eq[f[a, i[a]], f[f[e, b], i[b]]]}$

$\texttt{eq[}\mathbf{\texttt{f[a, i[a]]}}\texttt{, f[f[e, b], i[b]]]} \underset{\mathbf{I}}{\longrightarrow} \texttt{eq[e, f[f[e, b], i[b]]]}$

$\texttt{eq[e, }\mathbf{\texttt{f[f[e, b], i[b]]}}\texttt{]} \underset{\mathbf{A}}{\longrightarrow} \texttt{eq[e, f[e, f[b, i[b]]]]}$

$\texttt{eq[e, f[e, }\mathbf{\texttt{f[b, i[b]]}}\texttt{]]} \underset{\mathbf{I}}{\longrightarrow} \texttt{eq[e, f[e, e]]}$

$\texttt{eq[e, }\mathbf{\texttt{f[e, e]}}\texttt{]} \underset{\mathbf{N}}{\longrightarrow} \texttt{eq[e, e]}$

Finally we have

$\texttt{eq[e, e]} \underset{\texttt{Eq}}{\longrightarrow} \texttt{True}$

Appendix 3

Find an expression *e* such that:

$-4.2 \underset{\texttt{fg}}{\longrightarrow} e$

Solution(s):

```
1.4
```

Justification:

Finally we have

$-4.2 \underset{\texttt{f1}}{\longrightarrow} 2.8$

and

$2.8 \underset{\texttt{g}}{\longrightarrow} 1.4$

Appendix 4

Find an expression *e* such that:

$-4.2 \underset{\text{fg}}{\longrightarrow} e$

Solution(s):

1.4

Justification:

$-4.2 \underset{\text{fg}}{\longrightarrow} ?$

which is

$-4.2 \underset{\text{f1}\circ\text{g}\mid\text{f2}\circ\text{g}}{\longrightarrow} ?$

because

$\text{fg} \overset{\text{def}}{\longmapsto} \text{f} \circ \text{g} \overset{\text{def}}{\longmapsto} (\text{f1} \mid \text{f2}) \circ \text{g} \overset{\text{def}}{\longmapsto} \text{f1} \circ \text{g} \mid \text{f2} \circ \text{g}$

Then

$-4.2 \underset{\text{f1}\circ\text{g}}{\longrightarrow} ?$

Finally we have

$-4.2 \underset{\text{f1}}{\longrightarrow} 2.8$

and

$2.8 \underset{\text{g}}{\longrightarrow} 1.4$

□