Abella: A tutorial

Andrew Gacek

Rockwell Collins, Inc.
Trusted Systems Group

`http://abella.cs.umn.edu`

Motivation

We are interested in a framework for developing *formal systems*

Some example formal systems:

- Programming language (evaluation, typing, ...)
- Logic (provability, equivalences, ...)
- Process calculi (equivalences, behaviors, ...)

A framework should support:

- Specification, prototyping, reasoning
- Working with objects with variable binding structure

Notable Developments

- Alberto Momigliano: Howe's Method
- Todd Wilson: Results from *Practical Foundations for Programming Languages* by Robert Harper
- Alwen Tiu and Dale Miller: π-calculus meta-theory
- Andrew Gacek: Girard's proof of strong normalization for the simply-typed λ-calculus
- "Andrew Gacek": POPLmark Challenge

Key Features of Abella

- Higher-order abstract syntax
- Definitions with induction and co-induction
- The ∇-quantifier
- A two-level logic approach to reasoning

The λ-calculus

$$\frac{}{(\lambda x : a.\ r) \Downarrow (\lambda x : a.\ r)} \qquad \frac{m \Downarrow (\lambda x : a.\ r) \quad r[x := n] \Downarrow v}{(m\ n) \Downarrow v}$$

$$\frac{x : a \in \Gamma}{\Gamma \vdash x : a} \qquad \frac{\Gamma \vdash m : a \to b \quad \Gamma \vdash n : a}{\Gamma \vdash (m\ n) : b}$$

$$\frac{\Gamma, x : a \vdash r : b}{\Gamma \vdash (\lambda x : a.\ r) : a \to b}\ x \notin dom(\Gamma)$$

Subject Reduction

Theorem

If $e \Downarrow v$ *and* $\vdash e : t$ *then* $\vdash v : t$

Proof.

Induction on the judgment $e \Downarrow v$. □

Lemma

If $\Gamma, x : a \vdash r : t$ *and* $\Gamma \vdash n : a$ *then* $\Gamma \vdash r[x := n] : t$

Lemma

If $\Gamma_1 \vdash e : t$ *and* Γ_2 *is a permutation of* Γ_1 *then* $\Gamma_2 \vdash e : t$.

λ-caluclus Signature

```
sig demo.

kind     tm, ty     type.

type     app        tm -> tm -> tm.
type     abs        ty -> (tm -> tm) -> tm.

type     i          ty.
type     arrow      ty -> ty -> ty.

type     of         tm -> ty -> o.
type     eval       tm -> tm -> o.
```

Example

$\lambda x : (i \to i).\ \lambda y : i.\ (x\ y)$

```
abs (arrow i i) (x\ abs i (y\ app x y))
```

λ-caluclus Specification

```
module demo.

eval (abs A R) (abs A R).
eval (app M N) V :- eval M (abs A R), eval (R N) V.

of (app M N) B :- of M (arrow A B), of N A.
of (abs A R) (arrow A B) :- pi x\ (of x A => of (R x) B).
```

```
Specification "demo".

Theorem subject_reduction :  forall E V T,
   {eval E V} -> {of E T} -> {of V T}.


  ============================
   forall E V T, {eval E V} -> {of E T} -> {of V T}
```

```
  ============================
   forall E V T, {eval E V} -> {of E T} -> {of V T}

subject_reduction < induction on 1.

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  ============================
   forall E V T, {eval E V}@ -> {of E T} -> {of V T}
```

```
  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  ============================
   forall E V T, {eval E V}@ -> {of E T} -> {of V T}

subject_reduction < intros.

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H1 : {eval E V}@
  H2 : {of E T}
  ============================
   {of V T}
```

```
  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H1 : {eval E V}@
  H2 : {of E T}
  ============================
   {of V T}

subject_reduction < case H1.
Subgoal 1:

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H2 : {of (abs A R) T}
  ============================
   {of (abs A R) T}

Subgoal 2 is:
 {of V T}
```

```
Subgoal 1:

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H2 : {of (abs A R) T}
  ============================
   {of (abs A R) T}

Subgoal 2 is:
 {of V T}

subject_reduction < search.
Subgoal 2:

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H2 : {of (app M N) T}
  H3 : {eval M (abs A R)}*
  H4 : {eval (R N) V}*
  ============================
   {of V T}
```

```
Subgoal 2:

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H2 : {of (app M N) T}
  H3 : {eval M (abs A R)}*
  H4 : {eval (R N) V}*
  ============================
   {of V T}

subject_reduction < case H2.

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H3 : {eval M (abs A R)}*
  H4 : {eval (R N) V}*
  H5 : {of M (arrow A1 T)}
  H6 : {of N A1}
  ============================
   {of V T}
```

```
  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H3 : {eval M (abs A R)}*
  H4 : {eval (R N) V}*
  H5 : {of M (arrow A1 T)}
  ============================
   {of V T}

subject_reduction < apply IH to H3 H5.

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H4 : {eval (R N) V}*
  H6 : {of N A1}
  H7 : {of (abs A R) (arrow A1 T)}
  ============================
   {of V T}
```

```
  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H4 : {eval (R N) V}*
  H6 : {of N A1}
  H7 : {of (abs A R) (arrow A1 T)}
  ============================
   {of V T}

subject_reduction < case H7.

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H4 : {eval (R N) V}*
  H6 : {of N A1}
  H8 : {of n1 A1 |- of (R n1) T}
  ============================
   {of V T}
```

```
  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H4 : {eval (R N) V}*
  H6 : {of N A1}
  H8 : {of n1 A1 |- of (R n1) T}
  ============================
   {of V T}

subject_reduction < inst H8 with n1 = N.

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H4 : {eval (R N) V}*
  H6 : {of N A1}
  H9 : {of N A1 |- of (R N) T}
  ============================
   {of V T}
```

```
  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H4 : {eval (R N) V}*
  H6 : {of N A1}
  H9 : {of N A1 |- of (R N) T}
  ============================
   {of V T}

subject_reduction < cut H9 with H6.

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H4 : {eval (R N) V}*
  H10 : {of (R N) T}
  ============================
   {of V T}
```

```
  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H4 : {eval (R N) V}*
  H10 : {of (R N) T}
  ============================
   {of V T}

subject_reduction < apply IH to H4 H10.

  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H11 : {of V T}
  ============================
   {of V T}
```

```
  IH : forall E V T, {eval E V}* -> {of E T} -> {of V T}
  H11 : {of V T}
  ============================
   {of V T}

subject_reduction < search.
Proof completed.
```

Summary

```
Specification "demo".

Theorem subject_reduction : forall E V T,
  {eval E V} -> {of E T} -> {of V T}.

induction on 1.
intros.
case H1.

search.

case H2.
apply IH to H3 H5.
case H7.
inst H8 with n1 = N.
cut H9 with H6.
apply IH to H4 H10.
search.
```

Type Uniqueness

Theorem

If $\vdash e : t_1$ *and* $\vdash e : t_2$ *then* $t_1 = t_2$

"Lemma"

If $\Gamma \vdash e : t_1$ and $\Gamma \vdash e : t_2$ then $t_1 = t_2$

Lemma

If Γ *ctx and* $\Gamma \vdash e : t_1$ *and* $\Gamma \vdash e : t_2$ *then* $t_1 = t_2$

$$\frac{}{\cdot \ ctx} \qquad\qquad \frac{\Gamma \ ctx}{\Gamma, x : a \ ctx} \ x \notin dom(\Gamma)$$

Contexts and Judgments

```
Type     nil          olist.
Type     ::           o -> olist -> olist.

Type     { |- }       olist -> o -> prop.
```

```
Define member : o -> olist -> prop by
  member A (A :: L);
  member A (B :: L) := member A L.
```

Contexts Wellformedness

$$\frac{}{\cdot\ ctx} \qquad\qquad \frac{\Gamma\ ctx}{\Gamma, x : a\ ctx}\ x \notin dom(\Gamma)$$

```
Define ctx : olist -> prop by
  ctx nil;
  nabla x, ctx (of x T :: L) := ctx L.
```

Preparations for Type Uniqueness

```
Theorem type_unique : forall L E T1 T2,
  ctx L -> {L |- of E T1} -> {L |- of E T2} -> T1 = T2.
```

Proof by induction and case analysis on `{L |- of E T1}`, with further case analysis on `{L |- of E T2}`

- `member (of E T1) L`
- `E = app M N`
- `E = abs A R`

- `member (of E T2) L`
- `E = app M N`
- `E = abs A R`

Context Lemmas

```
Theorem ctx_member_unique : forall L E T1 T2,
  ctx L -> member (of E T1) L ->
           member (of E T2) L -> T1 = T2.
```

```
Theorem ctx_member_app_absurd : forall M N T L,
  ctx L -> member (of (app M N) T) L -> false.
```

```
Theorem ctx_member_abs_absurd : forall A R T L,
  ctx L -> member (of (abs A R) T) L -> false.
```

Base Case Lemma

```
Theorem ctx_member_unique : forall L E T1 T2,
  ctx L -> member (of E T1) L ->
           member (of E T2) L -> T1 = T2.
```

- `L = of E T1 :: L'`
- `L = B :: L'` and

  `member (of E T1) L'`

- `L = of E T2 :: L'`
- `L = B :: L'` and

  `member (of E T2) L'`

```
Define ctx : olist -> prop by
  ctx nil;
  nabla x, ctx (of x T :: L) := ctx L.
```

```
Theorem member_variable_absurd : forall L T, nabla x,
  member (of x T) L -> false.
```

```
  ============================
   forall L T, nabla x, member (of x T) L -> false

member_variable_absurd < induction on 1.

  IH : forall L T, nabla x, member (of x T) L * -> false
  ============================
   forall L T, nabla x, member (of x T) L @ -> false
```

```
  IH : forall L T, nabla x, member (of x T) L * -> false
  ============================
   forall L T, nabla x, member (of x T) L @ -> false

member_variable_absurd < intros.

  IH : forall L T, nabla x, member (of x T) L * -> false
  H1 : member (of n1 T) L @
  ============================
   false
```

```
  IH : forall L T, nabla x, member (of x T) L * -> false
  H1 : member (of n1 T) L @
  ============================
   false

member_variable_absurd < case H1.

  IH : forall L T, nabla x, member (of x T) L * -> false
  H2 : member (of n1 T) L3 *
  ============================
   false
```

```
  IH : forall L T, nabla x, member (of x T) L * -> false
  H2 : member (of n1 T) L3 *
  ============================
   false

member_variable_absurd < apply IH to H2.
Proof completed.
```

Base Case Lemma

```
Theorem ctx_member_unique : forall L E T1 T2,
  ctx L -> member (of E T1) L ->
           member (of E T2) L -> T1 = T2.
```

```
  ============================
   forall L E T1 T2, ctx L -> member (of E T1) L ->
     member (of E T2) L -> T1 = T2

ctx_member_unique < induction on 2.

  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  ============================
   forall L E T1 T2, ctx L -> member (of E T1) L @ ->
     member (of E T2) L -> T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  ============================
   forall L E T1 T2, ctx L -> member (of E T1) L @ ->
     member (of E T2) L -> T1 = T2

ctx_member_unique < intros.

  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx L
  H2 : member (of E T1) L @
  H3 : member (of E T2) L
  ============================
   T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx L
  H2 : member (of E T1) L @
  H3 : member (of E T2) L
  ============================
   T1 = T2

ctx_member_unique < case H2.
Subgoal 1:

  Variables: L, E, T1, T2, L1
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (of E T1 :: L1)
  H3 : member (of E T2) (of E T1 :: L1)
  ============================
   T1 = T2

Subgoal 2 is:
 T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (of E T1 :: L1)
  H3 : member (of E T2) (of E T1 :: L1)
  ============================
   T1 = T2

ctx_member_unique < case H3.
Subgoal 1.1:

  Variables: L, E, T1, T2, L1
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (of E T1 :: L1)
  ============================
   T1 = T1

Subgoal 1.2 is:
 T1 = T2

Subgoal 2 is:
 T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (of E T1 :: L1)
  ============================
   T1 = T1

ctx_member_unique < search.
Subgoal 1.2:

  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (of E T1 :: L1)
  H4 : member (of E T2) L1
  ============================
   T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (of E T1 :: L1)
  H4 : member (of E T2) L1
  ============================
   T1 = T2

ctx_member_unique < case H1.
Subgoal 1.2:

  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H4 : member (of n1 T2) L2
  H5 : ctx L2
  ============================
   T1 = T2

Subgoal 2 is:
 T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H4 : member (of n1 T2) L2
  H5 : ctx L2
  ============================
   T1 = T2

ctx_member_unique < apply member_variable_absurd to H4.
Subgoal 2:

  Variables: L, E, T1, T2, L1, B
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (B :: L1)
  H3 : member (of E T2) (B :: L1)
  H4 : member (of E T1) L1 *
  ============================
   T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (B :: L1)
  H3 : member (of E T2) (B :: L1)
  H4 : member (of E T1) L1 *
  ============================
   T1 = T2

ctx_member_unique < case H3.
Subgoal 2.1:

  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (of E T2 :: L1)
  H4 : member (of E T1) L1 *
  ============================
   T1 = T2

Subgoal 2.2 is:
 T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (of E T2 :: L1)
  H4 : member (of E T1) L1 *
  ============================
   T1 = T2

ctx_member_unique < case H1.
Subgoal 2.1:

  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H4 : member (of n1 T1) L2 *
  H5 : ctx L2
  ============================
   T1 = T2

Subgoal 2.2 is:
 T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H4 : member (of n1 T1) L2 *
  H5 : ctx L2
  ============================
   T1 = T2

ctx_member_unique < apply member_variable_absurd to H4.
Subgoal 2.2:

  Variables: L, E, T1, T2, L1, B
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (B :: L1)
  H4 : member (of E T1) L1 *
  H5 : member (of E T2) L1
  ============================
   T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H1 : ctx (B :: L1)
  H4 : member (of E T1) L1 *
  H5 : member (of E T2) L1
  ============================
   T1 = T2

ctx_member_unique < case H1.
Subgoal 2.2:

  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H4 : member (of (E n1) T1) L2 *
  H5 : member (of (E n1) T2) L2
  H6 : ctx L2
  ============================
   T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H4 : member (of (E n1) T1) L2 *
  H5 : member (of (E n1) T2) L2
  H6 : ctx L2
  ============================
   T1 = T2

ctx_member_unique < apply IH to H6 H4 H5.
Subgoal 2.2:

  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H4 : member (of (E n1) T2) L2 *
  H5 : member (of (E n1) T2) L2
  H6 : ctx L2
  ============================
   T2 = T2
```

```
  IH : forall L E T1 T2, ctx L -> member (of E T1) L * ->
         member (of E T2) L -> T1 = T2
  H4 : member (of (E n1) T2) L2 *
  H5 : member (of (E n1) T2) L2
  H6 : ctx L2
  ============================
   T2 = T2

ctx_member_unique < search.
Proof completed.
```

Context Lemmas

```
Theorem ctx_member_app_absurd : forall M N T L,
  ctx L -> member (of (app M N) T) L -> false.
```

```
Theorem ctx_member_abs_absurd : forall A R T L,
  ctx L -> member (of (abs A R) T) L -> false.
```

```
  ============================
   forall M N T L, ctx L -> member (of (app M N) T) L ->
     false

ctx_member_app_absurd < induction on 2.

  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  ============================
   forall M N T L, ctx L -> member (of (app M N) T) L @ ->
     false
```

```
  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  ============================
   forall M N T L, ctx L -> member (of (app M N) T) L @ ->
     false

ctx_member_app_absurd < intros.

  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  H1 : ctx L
  H2 : member (of (app M N) T) L @
  ============================
   false
```

```
  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  H1 : ctx L
  H2 : member (of (app M N) T) L @
  ============================
   false

ctx_member_app_absurd < case H2.
Subgoal 1:

  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  H1 : ctx (of (app M N) T :: L1)
  ============================
   false

Subgoal 2 is:
 false
```

```
  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  H1 : ctx (of (app M N) T :: L1)
  ============================
   false

ctx_member_app_absurd < case H1.
Subgoal 2:

  Variables: M, N, T, L, L1, B
  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  H1 : ctx (B :: L1)
  H3 : member (of (app M N) T) L1 *
  ============================
   false
```

```
  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  H1 : ctx (B :: L1)
  H3 : member (of (app M N) T) L1 *
  ============================
   false

ctx_member_app_absurd < case H1.
Subgoal 2:

  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  H3 : member (of (app (M n1) (N n1)) T) L2 *
  H4 : ctx L2
  ============================
   false
```

```
  IH : forall M N T L, ctx L ->
         member (of (app M N) T) L * -> false
  H3 : member (of (app (M n1) (N n1)) T) L2 *
  H4 : ctx L2
  ============================
   false

ctx_member_app_absurd < apply IH to H4 H3.
Proof completed.
```

Type Uniqueness

```
Theorem type_unique : forall L E T1 T2,
  ctx L -> {L |- of E T1} -> {L |- of E T2} -> T1 = T2.
```

```
  ============================
   forall L E T1 T2, ctx L -> {L |- of E T1} ->
     {L |- of E T2} -> T1 = T2

type_unique < induction on 2.

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  ============================
   forall L E T1 T2, ctx L -> {L |- of E T1}@ ->
     {L |- of E T2} -> T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  ============================
   forall L E T1 T2, ctx L -> {L |- of E T1}@ ->
     {L |- of E T2} -> T1 = T2

type_unique < intros.

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H2 : {L |- of E T1}@
  H3 : {L |- of E T2}
  ============================
   T1 = T2
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H2 : {L |- of E T1}@
  H3 : {L |- of E T2}
  ============================
   T1 = T2

type_unique < case H2.
Subgoal 1:

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H3 : {L |- of E T2}
  H4 : member (of E T1) L
  ============================
   T1 = T2

Subgoal 2 is:
  T1 = T2

...
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H3 : {L |- of E T2}
  H4 : member (of E T1) L
  ============================
   T1 = T2

type_unique < case H3.
Subgoal 1.1:

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : member (of E T1) L
  H5 : member (of E T2) L
  ============================
   T1 = T2

Subgoal 1.2 is:
 T1 = T2

...
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : member (of E T1) L
  H5 : member (of E T2) L
  ============================
   T1 = T2

type_unique < apply ctx_member_unique to H1 H4 H5.
Subgoal 1.1:

  Variables: L, E, T1, T2
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : member (of E T2) L
  H5 : member (of E T2) L
  ============================
   T2 = T2

Subgoal 1.2 is:
 T1 = T2

...
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : member (of E T2) L
  H5 : member (of E T2) L
  ============================
   T2 = T2

type_unique < search.
Subgoal 1.2:

  Variables: L, E, T1, T2, A, N, M
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : member (of (app M N) T1) L
  H5 : {L |- of M (arrow A T2)}
  H6 : {L |- of N A}
  ============================
   T1 = T2

Subgoal 1.3 is:
 T1 = arrow A B

...
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : member (of (app M N) T1) L
  H5 : {L |- of M (arrow A T2)}
  H6 : {L |- of N A}
  ============================
   T1 = T2

type_unique < apply ctx_member_app_absurd to H1 H4.
Subgoal 1.3:

  Variables: L, E, T1, T2, B, R, A
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : member (of (abs A R) T1) L
  H5 : {L, of n1 A |- of (R n1) B}
  ============================
   T1 = arrow A B

Subgoal 2 is:
 T1 = T2

...
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : member (of (abs A R) T1) L
  H5 : {L, of n1 A |- of (R n1) B}
  ============================
   T1 = arrow A B

type_unique < apply ctx_member_abs_absurd to H1 H4.
Subgoal 2:

  Variables: L, E, T1, T2, A, N, M
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H3 : {L |- of (app M N) T2}
  H4 : {L |- of M (arrow A T1)}*
  H5 : {L |- of N A}*
  ============================
   T1 = T2

Subgoal 3 is:
 arrow A B = T2

...
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H3 : {L |- of (app M N) T2}
  H4 : {L |- of M (arrow A T1)}*
  H5 : {L |- of N A}*
  ============================
   T1 = T2

type_unique < case H3.
Subgoal 2.1:

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L |- of M (arrow A T1)}*
  H5 : {L |- of N A}*
  H6 : member (of (app M N) T2) L
  ============================
   T1 = T2

Subgoal 2.2 is:
 T1 = T2

...
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L |- of M (arrow A T1)}*
  H5 : {L |- of N A}*
  H6 : member (of (app M N) T2) L
  ============================
   T1 = T2

type_unique < apply ctx_member_app_absurd to H1 H6.
Subgoal 2.2:

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L |- of M (arrow A T1)}*
  H5 : {L |- of N A}*
  H6 : {L |- of M (arrow A1 T2)}
  H7 : {L |- of N A1}
  ============================
   T1 = T2

Subgoal 3 is:
 arrow A B = T2
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L |- of M (arrow A T1)}*
  H5 : {L |- of N A}*
  H6 : {L |- of M (arrow A1 T2)}
  H7 : {L |- of N A1}
  ============================
   T1 = T2

type_unique < apply IH to H1 H4 H6.
Subgoal 2.2:

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L |- of M (arrow A1 T2)}*
  H5 : {L |- of N A1}*
  H6 : {L |- of M (arrow A1 T2)}
  H7 : {L |- of N A1}
  ============================
   T2 = T2

Subgoal 3 is:
 arrow A B = T2
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L |- of M (arrow A1 T2)}*
  H5 : {L |- of N A1}*
  H6 : {L |- of M (arrow A1 T2)}
  H7 : {L |- of N A1}
  ============================
   T2 = T2

type_unique < search.
Subgoal 3:

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H3 : {L |- of (abs A R) T2}
  H4 : {L, of n1 A |- of (R n1) B}*
  ============================
   arrow A B = T2
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H3 : {L |- of (abs A R) T2}
  H4 : {L, of n1 A |- of (R n1) B}*
  ============================
   arrow A B = T2

type_unique < case H3.
Subgoal 3.1:

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L, of n1 A |- of (R n1) B}*
  H5 : member (of (abs A R) T2) L
  ============================
   arrow A B = T2

Subgoal 3.2 is:
 arrow A B = arrow A B1
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L, of n1 A |- of (R n1) B}*
  H5 : member (of (abs A R) T2) L
  ============================
   arrow A B = T2

type_unique < apply ctx_member_abs_absurd to H1 H5.
Subgoal 3.2:

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L, of n1 A |- of (R n1) B}*
  H5 : {L, of n1 A |- of (R n1) B1}
  ============================
   arrow A B = arrow A B1
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L, of n1 A |- of (R n1) B}*
  H5 : {L, of n1 A |- of (R n1) B1}
  ============================
   arrow A B = arrow A B1

type_unique < apply IH to _ H4 H5.
Subgoal 3.2:

  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L, of n1 A |- of (R n1) B1}*
  H5 : {L, of n1 A |- of (R n1) B1}
  ============================
   arrow A B1 = arrow A B1
```

```
  IH : forall L E T1 T2, ctx L -> {L |- of E T1}* ->
         {L |- of E T2} -> T1 = T2
  H1 : ctx L
  H4 : {L, of n1 A |- of (R n1) B1}*
  H5 : {L, of n1 A |- of (R n1) B1}
  ============================
   arrow A B1 = arrow A B1

type_unique < search.
Proof completed.
```

Summary

```
Theorem type_unique : forall L E T1 T2,
  ctx L -> {L |- of E T1} -> {L |- of E T2} -> T1 = T2.

induction on 2. intros. case H2.
  case H3.
    apply ctx_member_unique to H1 H4 H5. search.
    apply ctx_member_app_absurd to H1 H4.
    apply ctx_member_abs_absurd to H1 H4.
  case H3.
    apply ctx_member_app_absurd to H1 H6.
    apply IH to H1 H4 H6. search.
  case H3.
    apply ctx_member_abs_absurd to H1 H5.
    apply IH to _ H4 H5. search.
```

Reasoning about Logic

```
sig cut.

kind    i, formula              type.

type    top, bot                formula.
type    p                       i -> formula.
type    and, or, imp            formula -> formula -> formula.
type    all, ex                 (i -> formula) -> formula.

type    form, hyp, conc         formula -> o.
```

Reasoning about Logic

```
module cut.

conc A :- hyp A.
conc top.
conc C :- hyp bot.

conc (and A B) :- conc A, conc B.
conc C :- hyp (and A B), hyp A => hyp B => conc C.

conc (or A B) :- conc A.
conc (or A B) :- conc B.
conc C :- hyp (or A B), hyp A => conc C, hyp B => conc C.

conc (imp A B) :- hyp A => conc B.
conc C :- hyp (imp A B), conc A, hyp B => conc C.

conc (all A) :- pi x\ conc (A x).
conc C :- hyp (all A), hyp (A T) => conc C.

conc (ex A) :- conc (A T).
conc C :- hyp (ex A), pi x\ hyp (A x) => conc C.
```

Reasoning about Logic

```
Define ctx : olist -> prop by
  ctx nil ;
  ctx (hyp A :: L) := ctx L.

Theorem cut_admissibility : forall L K C,
  {form K} -> ctx L ->
    {L |- conc K} -> {L, hyp K |- conc C} -> {L |- conc C}.
```

The proof is by nested induction on

- The size of the cut formula K: `{form K}`
- The height of `{L, hyp K |- conc C}`

Co-induction

```
CoDefine diverge : tm -> prop by
  diverge (app M N) := diverge M ;
  diverge (app M N) :=
    exists A R, {eval M (abs A R)} /\ diverge (R N).
```

```
Theorem omega_diverge :
  diverge (app (abs i (x\ app x x)) (abs i (x\ app x x))).
```

```
Theorem eval_diverge_absurd : forall M V,
  {eval M V} -> diverge M -> false.
```

Strong Normalization for Simply-typed λ-calculus

```
step (app M N) (app M' N) :- step M M'.
step (app M N) (app M N') :- step N N'.
step (app (abs A R) M) (R M).
step (abs A R) (abs A R') :- pi x\ step (R x) (R' x).
```

```
Define sn : tm -> prop by
  sn M := forall N, {step M N} -> sn N.
```

```
Define reduce : tm -> ty -> prop by
  reduce M i := {of M i} /\ sn M ;
  reduce M (arrow A B) :=
    {of M (arrow A B)} /\
    (forall U, reduce U A -> reduce (app M U) B).
```

```
Theorem strong_norm : forall M A, {of M A} -> sn M.
```

Going Forward

`http://abella.cs.umn.edu/`

- Walkthroughs
- Examples
- Downloads
- Documentation
- Papers

Programming with Higher Order Logic