The decomposition rules for $A \Rightarrow B$ and $\forall x.B(x)$ are the most difficult to motivate and use intuitively. Since the evidence for $A \Rightarrow B$ is a function $\lambda(x.b(x))$, a reader might expect to see a decomposition rule name such as $apply(f; a)$ or abbreviated to $ap(f; a)$. However, a Gentzen sequent-style proof rule for decomposing an implication has this form:

$H, f : A \Rightarrow B, H' \vdash G$ by *ImpL* on f

1. $H, f : A \Rightarrow B,\ H' \vdash A$
2. $H, f : A \Rightarrow B,\ v : B, H' \vdash G$

As the proof proceeds, the two subgoals 1 and 2 with conclusions A and G respectively will be refined, say with proof terms a and $g(f, v)$ respectively. We need to indicate that the value v is $ap(f; a)$, but at the point where the rule is applied, we only have slots for these subterms and a name v for the new hypothesis B. So the rule form is $apseq(f; slot_a; v.slot_g(v))$ where we know that v will be assigned the value $ap(f; slot_a)$ to "sequence" the two subgoals properly. So *apseq* is a sequencing operator as well as an application, and when the subterms are created, we can evaluate the term further as we show below. We thus express the rule as follows.

$H, f : A \Rightarrow B,\ H' \vdash G$ *by* $apseq(f; slot_a; v.slot_g(v))$

$H, f : A \Rightarrow B,\ v : B, H' \vdash G$ *by* $slot_g(v)$

$H, f : A \Rightarrow B,\ H' \vdash A$ *by* $slot_a$

We evaluate the term $apseq(f; a; v.g(v))$ to $g(ap(f; a))$ or more succinctly to $g(f(a))$. This simplification can only be done on the final bottom up pass of creating a closed proof expression, one with no slots.

Semantic consistency. With this introduction, the following rules should make sense. They define what we will call the *pure proof expressions*. It should also be clear that we can easily prove by induction on the structure of proofs that there is computational evidence for every provable formula and that the evidence is polymorphic (uniform). This provides a simple semantic consistency proof for *iFOL* and easy demonstrations that specific formulas such as $P \vee \sim P$ are not provable.

2.2. First-order refinement style proof rules over domain of discourse D

Minimal Logic
Construction rules

- **And Construction**

  $H \vdash A\&B$ *by* $pair(slot_a; slot_b)$
  $H \vdash A$ *by* $slot_a$
  $H \vdash B$ *by* $slot_b$

- **Exists Construction**

  $H \vdash \exists x.B(x)$ *by* $pair(d; slot_b(d))$
  $H \vdash d \in D$ *by* $obj(d)$
  $H \vdash B(d)$ *by* $slot_b(d)$

- **Implication Construction**

  $H \vdash A \Rightarrow B$ *by* $\lambda(x.slot_b(x))$ *new* x
  $H, x : A \vdash B$ *by* $slot_b(x)$

- **All Construction**

  $H \vdash \forall x.B(x)$ *by* $\lambda(x.slot_b(x))$ *new* x
  $H, x : D \vdash B(x)$ *by* $slot_b(x)$

- **Or Construction**

  $H \vdash A \vee B$ *by* $inl(slot_l)$
  $H \vdash A$ *by* $slot_l$

  $H \vdash A \vee B$ *by* $inr(slot_r)$
  $H \vdash B$ *by* $slot_r$

Decomposition rules

- **And Decomposition**

  $H, x : A\&B,\ H' \vdash G$ *by* $spread(x; l, r.slot_g(l, r))$ *new* l, r
  $H,\ l : A,\ r : B,\ H' \vdash G$ *by* $slot_g(l, r)$

- **Exists Decomposition**

  $H, x : \exists y.B(y),\ H' \vdash G$ *by* $spread(x; d, r.slot_g(d, r))$ *new* d, r
  $H,\ d : D,\ r : B(d),\ H' \vdash G$ *by* $slot_g(d, r)$

- **Implication Decomposition**

  $H, f : A \Rightarrow B,\ H' \vdash G$ *by* $apseq(f; slot_a; v.slot_g[ap(f; slot_a)/v])$ *new* v[9]
  $H, f : A \Rightarrow B,\ H' \vdash A$ *by* $slot_a$
  $H, f : A \Rightarrow B,\ H',\ v : B \vdash G$ *by* $slot_g(v)$

- **All Decomposition**

  $H, f : \forall x.B(x),\ H' \vdash G$ *by* $apseq(f; d; v.slot_g[ap(f; d)/v])$
  $H, f : \forall x.B(x),\ H' \vdash d \in D$ *by* $obj(d)$
  $H, f : \forall x.B(x),\ H', v : B(d) \vdash G$ *by* $slot_g(v)$[10]

- **Or Decomposition**

  $H, y : A \vee B,\ H' \vdash G$ *by* $decide(y; l.leftslot(l);\ r.rightslot(r))$
  $H, l : A, H' \vdash G$ *by* $leftslot(l)$
  $H, r : B, H' \vdash G$ *by* $rightslot(r)$

- **Hypothesis**

  $H, d : D,\ H' \vdash d \in D$ *by* $obj(d)$

  $H, x : A,\ H' \vdash A$ *by* $hyp(x)$

  We usually abbreviate the justifications to *by* d and *by* x respectively.

[9] This notation shows that $ap(f; slot_a)$ is substituted for v in $g(v)$. In the *CTT* logic we stipulate in the rule that $v = ap(f; slot_a)$ *in* B.
[10] In the *CTT* logic, we use equality to stipulate that $v = ap(f; d)$ *in* $B(v)$ just before the hypothesis $v : B(d)$.