Kruskal’s tree theorem in Type Theory

Dominique Larchey-Wendling
TYPES team

LORIA – CNRS
Nancy, France

http://www.loria.fr/~larchey/Kruskal

Foundations for the Practical Formalization of Mathematics 2017
Well Quasi Orders (WQO) 1/2

- Important concept in Computer Science:
  - strengthens well-foundedness, more stable
  - termination of rewriting (Dershowitz, RPO)
  - size-change termination, terminator (Vytiniostis, Coquand ...)

- Important concept in Mathematics:
  - Dickson’s lemma, Higman’s lemma
  - Higman’s theorem, Kruskal’s theorem
  - Robertson-Seymour theorem (graph minor theorem)
  - Unprovability result: Kruskal theorem not in PA (Friedman)
Well Quasi Orders (WQO) 2/2

• for \( \leq \) a quasi order over \( X \): reflexive & transitive binary relation

• several classically equivalent definitions (see e.g. JGL 2013)
  – almost full: each \((x_i)_{i \in \mathbb{N}}\) has a good pair \((x_i \leq x_j \text{ with } i < j)\)
  – \(\leq\) well-founded and no \(\infty\) antichain
  – finite basis: \(U = \uparrow U\) implies \(U = \uparrow F\) for some finite \(F\)
  – \(\downarrow U \mid U \subseteq X\) well-founded by \(\subset\)

• many of these equivalences do not hold intuitionistically
WQOs are stable under type constructs

- Given a WQO $\leq$ on $X$, we can lift $\leq$ to WQOs on:
  
  **Higman lemma**: $\text{list}(X)$ with subword($\leq$)
  
  **Higman thm**: $\text{btree}(k, X)$ with $\text{emb\_product}(\leq)$ (any $k \in \mathbb{N}$)
  
  **Kruskal theorem**: $\text{tree}(X)$ with $\text{emb\_homeo}(\leq)$

- These theorem are *closure properties* of the class of WQOs

- Other noticable results:
  
  **Dickson’s lemma**: $(\mathbb{N}^k, \leq)$ is a WQO
  
  **Finite sequence thm**: $\text{list}(\mathbb{N})$ WQO under subword($\leq$)
  
  **Ramsey theorem**: $\leq_1$ and $\leq_2$ WQOs imply $\leq_1 \times \leq_2$ WQO
What Intuitionistic Kruskal Tree Theorem?

• The meaning of those closure theorems intuitionistically:
  – depends of what is a WQO (which definition?)
  – but not on e.g. \texttt{emb\_homeo} which has an inductive definition

• What is a suitable intuitionistic definition of WQO?
  – quasi-order does not play an important/difficult role
  – should be classically equivalent to the usual definition
  – should intuitionistically imply almost full
  – intuitionistic WQOs must be stable under liftings

• Allow the proof and use of Ramsey, Higman, Kruskal... results
Intuitionistic formulations of WQOs 1/2

• Almost full relations (Veldman&Bezem 93)
  – each \((x_i)_{i \in \mathbb{N}}\) has \(x_i R x_j\) with \(i < j\)
  – works for Higman and Kruskal theorems (Veldman 04)
  – uses *stumps* over \(\mathbb{N}\) which require *Brouwer’s thesis*

• Bar induction (Coquand&Fridlender 93)
  – *bar extends* \((\text{good } R) [\cdot]\)
  – works for the general Higman lemma (Fridlender 97)

• Well-foundedness (Seisenberger 2003)
  – \(\text{extends}^{(-1)}\) is well-founded on \(\text{Bad}(R)\)
  – works for Higman lemma and Kruskal theorem
  – requires *decidability* of \(R\)
Intuitionistic formulations of WQOs 2/2

- Almost full relations (Vytiniostis&Coquand&Wahlstedt 12)
  - \( \text{af}(R) \) inductively defined
  - works for Ramsey theorem
  - intuitionistically equivalent to \( \text{bar extends} (\text{good } R) \) [ ]

- Seisenberger’s definition not equiv. to Coquand&Fridlender for undecidable \( R \)

- Veldman&Bezem definition works for \( R \) over \( \mathbb{N} \) (not over arbitrary types) but requires Brouwer’s thesis

- Let us introduce
  - bar inductive predicates
  - Coquand et al. inductive definition of almost full
Bar inductive predicate, accessibility predicate (i)

- Given \( \mathcal{T} : X \to X \to \text{Prop} \), \( x : X \) and \( Q : X \to \text{Prop} \)
- \( x \) bars \( Q \) if every \( \infty \) path from \( x \) meets \( Q \)
- \( x \) is accessible if every \( \infty \) path from \( x \) meets \( \_ \mapsto \text{False} \)
- Inductive definitions (\text{Prop} or \text{Type}) are stronger (intui.)

\[
\frac{Q \ x}{\text{bar } \mathcal{T} \ Q \ x} \quad \frac{\forall y, \mathcal{T} \ x \ y \to \text{bar } \mathcal{T} \ Q \ y}{\text{bar } \mathcal{T} \ Q \ x} \quad \frac{\forall y, \mathcal{T} \ x \ y \to \text{acc } \mathcal{T} \ y}{\text{acc } \mathcal{T} \ x}
\]

- Axioms (like Brouwer’s bar thesis) for equivalence
- Obviously: \( \text{acc } \mathcal{T} \ x \) iff \( \text{bar } \mathcal{T} \ (\_ \mapsto \text{False}) \ x \)
FAN theorem and bar over lists (ii)

- **inductive FAN theorem:** \( \overline{\text{bar }} T \ Q \ x \rightarrow \overline{\text{bar }} T^\circ \ \forall Q \ [x] \)
  - for monotonic \( Q \): \( \forall x \ y, T \ x \ y \rightarrow Q \ x \rightarrow Q \ y \)
  - \( T^\circ \ l \ m \) iff \( \forall y, y \in m \rightarrow \exists x, x \in l \land T \ x \ y \) (direct image)
  - \( (\forall Q) \ l \) iff \( \forall x, x \in l \rightarrow Q \ x \) (finite quantification)

- **We use** \( \overline{\text{bar }} T \ Q \) with \( T = \text{extends} \) and \( Q = \text{good } R \)
  - \( \text{extends } l \ m \) iff \( m = _{-}:: l \)
  - \( \text{good } R \ ll \) iff \( ll = l ++ [b] :: m ++ [a] :: r \) for some \( a \ R \ b \)

- **bar extends (good } R) [] \)** iff
  
  \text{iterated extensions of [] must cross a good list}

- every infinite sequence contains a good pair (almost full)
Well-founded trees over a type $X$

- Well-founded trees $\text{wft}(X)$
  - branching indexed by $X$
  - the least fixpoint of $\text{wft}(X) = \{\star\} + X \to \text{wft}(X)$

- Given a branch $f : \mathbb{N} \to X$, compute its height:
  
  - $f(1 + \cdot) = x \mapsto f(1 + x)$
  - $\text{ht}(\text{inl } \star, \_ ) = 0$
  - $\text{ht}(\text{inr } g, f) = 1 + \text{ht}(g(f_0), f(1 + \cdot))$

- Veldman’s stumps are sets of branches of trees in $\text{wft}(\mathbb{N})$
Coquand’s Almost full relations, step by step

1. Veldman et al.: \( \forall f : \mathbb{N} \to X, \exists i < j, f_i R f_j \)

2. Logically eq. variant: \( \forall f : \mathbb{N} \to X, \exists n, \exists i < j < n, f_i R f_j \)

3. Partially informative: \( \forall f : \mathbb{N} \to X, \{ n \mid \exists i < j < n, f_i R f_j \} \)

4. Variant: \( \{ h : (\mathbb{N} \to X) \to \mathbb{N} \mid \forall f, \exists i < j < h(f), f_i R f_j \} \)

5. Variant: \( \{ t : \text{wft}(X) \mid \forall f, \exists i < j < \text{ht}(t, f), f_i R f_j \} \)

6. Coquand et al.: is defined as an inductive predicate \( \text{af}_t(R) \)
   - the prefix of length \( \text{ht}(t, f) \) of \( f : \mathbb{N} \to X \) contains a good pair
   - the computational content is (for every sequence \( f : \mathbb{N} \to X \)):
     - a bound on the size of the search space for good pairs
     - and it is not a good pair
A well-founded tree for \((\mathbb{N}, \leq)\)

- Property: \(\forall f : \mathbb{N} \to \mathbb{N}, \exists i < j < 2 + f_0, f_i \leq f_j\)

- In \(\text{wft}(\mathbb{N})\), we define \(T_n\) the tree of uniform height \(n\):
  - \(T_0 = \text{inl}(\ast)\) and \(T_{1+n} = \text{inr}(\_ \mapsto T_n)\)
  - for any \(f : \mathbb{N} \to \mathbb{N}\), \(\text{ht}(T_n, f) = n\)

- And \(T_\leq = \text{inr}(n \mapsto T_{1+n})\)

- Hence \(\text{ht}(T_\leq, f) = 1 + \text{ht}(T_{1+f_0}, f(1 + \cdot)) = 2 + f_0\)
Almost full relations, inductively

- Lifted relation: $x \ (R \uparrow u) \ y = x \ R \ y \lor u \ R \ x$
  - in $R \uparrow u$, elements above $u$ are forbidden in bad sequences

- **full**: $\text{rel}_2 X \rightarrow \text{Prop}$ and $\text{af}_t : \text{rel}_2 X \rightarrow \text{Type}$

\[
\begin{align*}
\forall x, y, x \ R \ y & \quad \text{full } R \\
\text{full } R & \quad \text{af}_t \ R \\
\forall u, \text{af}_t (R \uparrow u) & \quad \text{af}_t \ R
\end{align*}
\]

- $\text{af\_securedby} : \text{wft}(X) \rightarrow \text{rel}_2 X \rightarrow \text{Prop}$:
  - $\text{af\_securedby}(\text{inl } \star, R) = \text{full } R$
  - $\text{af\_securedby}(\text{inr } g, R) = \forall u, \text{af\_securedby}(g(u), R \uparrow u)$

- these are intuitionistically “equivalent” (hold in Type, not Prop):
  - $\text{af}_t \ R$ and $\{ t : \text{wft}(X) \ | \ \text{af\_securedby}(t, R) \}$
  - and $\{ t : \text{wft}(X) \ | \ \forall f, \exists i < j < \text{ht}(t, f), \ f_i \ R \ f_j \}$
Almost full relations, by bar inductive predicates

• good \( R : \text{list} \ X \to \text{Prop} \)
  
  – good \( R \ ll \) iff \( ll = l ++ \boxed{b} :: m ++ \boxed{a} :: r \) for some \( a R b \)
  
  – beware of the (implicit) use snoc lists
  
  – good has an easy inductive definition

• for \( P : \text{list} \ X \to \text{Prop} \), we define \( \text{bar}_t P : \text{list} \ X \to \text{Type} \)

\[
\frac{P \ ll}{\text{bar}_t P \ ll} \quad \frac{\forall u, \text{bar}_t P (u :: ll)}{\text{bar}_t P \ ll}
\]

• we show: \( \text{af}_t (R \uparrow a_n \uparrow \ldots \uparrow a_1) \) iff \( \text{bar}_t (\text{good} \ R) [a_1, \ldots, a_n] \)

• another characterization: \( \text{af}_t R \) iff \( \text{bar}_t (\text{good} \ R) [] \)
Almost full relations, some properties

- \texttt{af.t refl}: if \texttt{af.t} $R$ then $=_X \subseteq R$ (iff in case $X$ is finite)
- \texttt{af.t inc}: if $R \subseteq S$ and \texttt{af.t} $R$ then \texttt{af.t} $S$
- \texttt{af.t surjective} (easy but very useful):
  - for $f : X \rightarrow Y \rightarrow \text{Prop}$, $R : \text{rel}_2 X$ and $S : \text{rel}_2 Y$
  - if $f$ surjective: $\forall y, \{ x \mid f \ x \ y \}$
  - if $f$ morphism: $f \ x_1 \ y_1$ and $f \ x_2 \ y_2$ and $x_1 \ R \ x_2$ imply $y_1 \ S \ y_2$
  - then \texttt{af.t} $R$ implies \texttt{af.t} $S$
- Ramsey (Coquand): \texttt{af.t} $R$ and \texttt{af.t} $S$ imply \texttt{af.t}($R \cap S$)
  - he deduces \texttt{af.t}($R \times S$) and \texttt{af.t}($R + S$)
- \textit{I stop because you may be almost full (but it is a MUST READ)}
Higman lemma and the subword relation

- Given $R : \text{rel}_2 X$ over a type $X$
- The subword relation $<^w_R : \text{rel}_2 (\text{list} X)$ defined by 3 rules

\[
\begin{align*}
[] & <^w_R [] \\
l & <^w_R m \\
a R b & l <^w_R m \\
& a :: l <^w_R b :: m
\end{align*}
\]

- also write subword $R$ for $<^w_R$
- Higman lemma (Fridlender 97, non informative version):

\[
\text{bar} (\text{good} R) [\;] \quad \text{implies} \quad \text{bar} (\text{good} (\text{subword} R)) [\;]
\]

- Nearly the same proof works for \text{bar}_t instead of \text{bar}
- But this proof cannot be generalized to finite trees...
The product tree embedding, Higman theorem

- trees with same type for all arities: \( \text{tree } X = X \times \text{list}(\text{tree } X) \)
- trees of breadth bounded by \( k \in \mathbb{N} \):
  \[
  \text{btree } k \ X = \{ t \mid \text{tree}_\text{fall}(\langle -|ll \rangle \mapsto \text{length} \ ll < k) \ t \}
  \]
- any \( t \in T \) is \( t = \langle x|t_1,\ldots,t_n \rangle \) with \( n < k, x \in X \) and \( t_i \in T \)
- for a relation \( R : \text{rel}_2 X \), we define (needs some work...)
  \[
  s \prec^X_R t_i \quad \frac{s \prec^X_R \langle x_n|t_1,\ldots,t_n \rangle}{x R y \quad s_1 \prec^X_R t_1,\ldots,s_n \prec^X_R t_n \quad \langle x|s_1,\ldots,s_n \rangle \prec^X_R \langle y|t_1,\ldots,t_n \rangle}
  \]
- also write \( \text{emb}_\text{tree}_\text{product} \ R \) for \( \prec^X_R \)
- Higman theorem: \( \text{af}_\text{t} \ R \) implies \( \text{af}_\text{t}(\prec^X_R) \) on \( \text{btree } k \ X \)
The homeomorphic embedding, Krukal theorem

- one type $X$ for all arities: $\text{tree } X = X \times \text{list(tree } X)$
- for $R : \text{rel}_2 X$, we define $<^*_R$ by nested induction

\[
\frac{s <^*_R t_i}{s <^*_R \langle x_n|t_1, \ldots, t_n \rangle}
\]

\[
x_i \ R \ x_j \ [s_1, \ldots, s_i] (\text{subword } <^*_R) [t_1, \ldots, t_j]
\]

\[
\langle x_i|s_1, \ldots, s_i \rangle <^*_R \langle x_j|t_1, \ldots, t_j \rangle
\]

- hand-written elimination scheme (nested induction)
- we also write $\text{emb}_{\text{tree homeo}} R$ for $<^*_R$
- Kruskal theorem: $\text{af}_t R$ implies $\text{af}_t (<^*_R)$
Plan of the rest of the presentation

- high level and informal proof principles of Higman’s theorem
  - with ideas from Veldman (mostly), Fridlender and Coquand
  - \texttt{tree}(X_n)_{n<k}, one type (and one relation) for each arity

- focus on several implementation challenges of that proof
  - \texttt{tree}(X_n) as a (decidable) subtype of \texttt{tree}(\sum X_n)
  - embed \( \sum X_n \) in a (specialized) universe \( U \)
  - empty type grounded induction for \( \texttt{af\_t}, \ldots \)

- what about the non-informative case \( \texttt{af} \)?
  - beware \( \texttt{af} R \) is weaker than \texttt{inhabited}(\texttt{af\_t} R)
  - well-foundedness up to a projection

- from Higman theorem to Kruskal theorem (remarks)
The product tree embedding, Higman theorem

• \( \text{tree}(X_n)_{n<k} = T \) where \( T \) is lfp of \( T = \sum_{n=0}^{k-1} X_n \times T^n \)

• one type \( X_n \) for each arity \( n < k \)

• any \( t \in T \) is \( t = \langle x_n|t_1, \ldots, t_n \rangle \) with \( x_n \in X_n \) and \( t_i \in T \)

• for arity-indexed relations \( R : \forall n < k, rel_2(X_n) \), we define

\[
\begin{align*}
    s &<^h_R t_i \\
    s &<^h_R \langle x_n|t_1, \ldots, t_n \rangle \\
    x_n R_n y_n &<^h_R t_1, \ldots, s_n <^h_R t_n \\
    \langle x_n|s_1, \ldots, s_n \rangle &<^h_R \langle y_n|t_1, \ldots, t_n \rangle
\end{align*}
\]

• also write \( \text{emb} \_\text{tree} \_\text{higman} \) \( R \) for \( <^h_R \)

• Higman thm.: \( (\forall n < k, \text{af} \_\text{t} R_n) \) implies \( \text{af} \_\text{t}(<^h_R) \)
Higman theorem, based on (Veldman 2004)

- each \( \text{af}_t R_n \) is witnessed by \( w_n : \text{af}_t \text{securedby}(w_n, R_n) \)
- easier outermost induction on \([w_0, \ldots, w_{k-1}]\) (lexicographic)
- apply rule 2, hence prove: \( \forall t, \text{af}_t (<^h_R \uparrow t) \)
- do this by structural induction on \( t \)
  - \( t = \langle x_i | t_1, \ldots, t_i \rangle \) with \( i < k \)
  - we can assume \( \text{af}_t (<^h_R \uparrow t_1), \ldots, \text{af}_t (<^h_R \uparrow t_i) \)
  - we show \( \text{af}_t (<^h_R \uparrow \langle x_i | t_1, \ldots, t_i \rangle) \)
  - depends on \( i = 0 \) or not, \( w_i = \text{inl} \star \) or not
Higman thm, case of leaves \((i = 0 \text{ and } w_0 = \text{inr} \ g)\)

- we have \(t = \langle x_0 | \emptyset \rangle \) \((i = 0)\)

- \(R'_0 = R_0 \uparrow x_0\) is \(\mathsf{af} \cdot \mathsf{t}\), witnessed by \(w'_0 = g(x_0)\)

- \(R'_j = R_j\) and \(w'_j = w_j\) for \(0 < j < k\)
  - \(w'_0 = g(x_0)\) is a sub-\(\mathsf{wft}(X_0)\) of \(w_0 = \text{inr} \ g\), hence simpler
  - \([w'_0, w_1, \ldots, w_{k-1}]\) easier than \([w_0, w_1, \ldots, w_{k-1}]\)
  - we deduce \(\mathsf{af} \cdot \mathsf{t}(<^h_{R'})\) by induction

- we show \(<^h_{R'} \subseteq <^h_{R} \uparrow \langle x_0 | \emptyset \rangle\) (relatively easy to check)

- we conclude \(\mathsf{af} \cdot \mathsf{t}(<^h_{R} \uparrow \langle x_0 | \emptyset \rangle)\)
Higman thm, case of leaves ($i = 0$ and $w_0 = \text{inl} \star$)

- $t = \langle x_0 | \emptyset \rangle$
- $R_0 \uparrow x_0 = R_0$ because $R_0$ is (already) full ($w_0 = \text{inl} \star$)
- but then we have $x_0 \not\in R_0$ for any $y$
- hence we deduce $\langle x_0 | \emptyset \rangle <^h_R \langle x_j | v_1, \ldots, v_j \rangle$
  - any (finite) tree contains a leaf $\langle y | \emptyset \rangle$
  - $\langle x_0 | \emptyset \rangle$ embeds into any leaf, e.g. $\langle y | \emptyset \rangle$
- we deduce $<^h_R \uparrow \langle x_0 | \emptyset \rangle$ is full (trivial to check)
- we conclude $\text{af.t}(<^h_R \uparrow \langle x_0 | \emptyset \rangle)$
Higman thm (0 < i < k and \( w_i = \text{inr } g \)) 1/2

- let \( T = \text{tree}(X_0, \ldots, X_{k-1}) \)
- we have \( t = \langle x_i|t_1, \ldots, t_i \rangle \) with \( 0 < i < k \)
- \( X'_j = X_j \) and \( R'_j = R_j \) for \( j \not\in \{i - 1, i\} \)
- \( X'_i = X_i \) and \( R'_i = R_i \uparrow x_i \) is \( \text{af.t} \) for \( w'_i = g(x_i) \) simpler than \( w_i \)
- \( X'_{i-1} = X_{i-1} + \sum_{p=0}^{i-1} X_i \times T \) and \( R'_{i-1} = R_{i-1} + \sum_{p=0}^{i-1} R_i \times (\langle \text{h}_R \uparrow t_p \rangle) \)
  - \( R'_{i-1} \) is \( \text{af.t} \) by Ramsey, obtain \( w'_{i-1} \)
  - \([\ldots, w'_{i-1}, w'_i, \ldots]\) easier than \([\ldots, w_{i-1}, w_i, \ldots]\)
  - we deduce \( \text{af.t}(\langle \text{h}_R' \rangle) \) by induction
- we show \( \text{af.t}(\langle \text{h}_R' \rangle) \) implies \( \text{af.t}(\langle \text{h}_R \uparrow \langle x_i|t_1, \ldots, t_i \rangle \rangle) \) (not easy)
• with $X'_{i-1} = X_{i-1} + \sum_{p=0}^{i-1} X_i \times T$, define an evaluation map

• $ev: \text{tree}(X_0, \ldots, X'_{i-1}, X_i, \ldots) \rightarrow \text{tree}(X_0, \ldots, X_{k-1})$
  
  $ev(\langle y_j| t_1, \ldots, t_j \rangle) = \langle y_j| ev t_1, \ldots, ev t_j \rangle$ for $j \neq i-1$

  $ev(\langle y_{i-1}| t_1, \ldots, t_{i-1} \rangle) = \langle y_{i-1}| ev t_1, \ldots, ev t_{i-1} \rangle$

  $ev(\langle (p, y_i, t)| t_1, \ldots, t_{i-1} \rangle) = \langle y_i| \text{insert } t p [ev t_1, \ldots, ev t_{i-1}] \rangle$

• $ev$ (is surjective and) has finite inverse images
  
  allows the use of $\text{bar } t$ induction and the FAN theorem

• use $ev$ to show $\text{af } t(\langle^{R'}h \rangle)$ implies $\text{af } t(\langle^{R'}h \uparrow \langle x_i| t_1, \ldots, t_i \rangle)$
  
  combinatorial principle: $\forall x \in X, P_x \lor Q_x \Rightarrow \forall x P_x \lor \exists x Q_x$

  and more complex version (see later)

  very technical part of Coq proof (largely absent from paper)
Higman thm (0 < i < k and $w_i = \text{inl} \star$)

- $T = \text{tree}(X_0, \ldots, X_{k-1})$ and $t = \langle x_i | t_1, \ldots, t_i \rangle$ with $0 < i < k$
- $w_i = \text{inl} \star$ thus we have $R_i$ is full on $X_i$
- $X'_j$ and $R'_j$ for $j \neq i$ as in case $w_i = \text{inr} \; g$

- $X'_i = \emptyset$ with any $R'_i$ (only one exists) is af t

- ensure case where $X'_i = \emptyset$ is simpler than $R_i$ is full on $X_i$
  - $w'_i = \text{None}$ is simpler than $w_i = \text{Some}(\text{inl} \star)$
  - we deduce af t($<^h_{R'}$) by induction

- we show af t($<^h_{R'}$) implies af t($<^h_R \uparrow \langle x_i | t_1, \ldots, t_i \rangle$)
  - similar to the case $w_i = \text{inr} \; g$
  - but not easy to factorize the Coq duplicated code
Higman thm \((i < k \text{ and } w_i = \text{None})\)

- \(T = \text{tree}(X_0, \ldots, X_{k-1})\) and \(t = \langle x_i | t_1, \ldots, t_i \rangle\) with \(0 < i < k\)
- but because \(w_i = \text{None}\), we have \(X_i = \emptyset\)
- this contradicts \(x_i \in X_i\); an easy case indeed

The induction principle of Veldman’s proof

- lexicographic product (corresponds to nested induction)
- not grounded on full relations (witnessed by the empty \(\text{wft}\))
- but grounded on [empty types]
- empty types are sub-cases of full relations
Remarks on the implementation of that proof

- Implements “well” for e.g. at most unary/binary trees

Theorem higman_abt_t : forall Z T, @af_t Z T
  -> forall Y S, @af_t Y S
  -> forall X R, @af_t X R
  -> af_t (embed_abtree R S T).

Proof. do 3 (induction 1 using af_t_dep_rect); .... End.

- Thought it requires a dependent induction principle for af_t

- But that does not work for parameterized breadth $k$
  - $\text{tree}(X_n)_{n<k}$ VERY cumbersome to work with
  - $[\ldots, w'_{i-1}, w'_i, \ldots]$ “easier” than $[\ldots, w_{i-1}, w_i, \ldots]$
  - but the $w'_{i-1} : \text{wft} X'_{i-1}$ and $w_{i-1} : \text{wft} X_{i-1}$ not same type!!
A dependent induction principle for \texttt{af\_t}

Section \texttt{af\_t\_dep\_rect}.

Variable (P : forall X, relation X \rightarrow Type).

Hypothesis \texttt{HP0} : P \forall R.

Hypothesis \texttt{HP1} : forall X R, full R \rightarrow P \forall R \rightarrow \forall P X R.

Hypothesis \texttt{HP2} : forall X R, (forall x, \texttt{af\_t}(R rlift x))
\rightarrow (forall x, P (R rlift x))
\rightarrow \forall P X R.

Theorem \texttt{af\_t\_dep\_rect} : forall X R, \texttt{af\_t} R \rightarrow \forall P X R.

End \texttt{af\_t\_dep\_rect}.
Finite Trees in Coq

- Dependent types: nice way to represent complex data structures
- But too much dependency can make your life miserable
- Hence we represent $\text{tree}(X_n)_{n \in \mathbb{N}}$ by:

$$\{ t : \text{tree}(\sum X_n) \mid \text{tree\_fall} (x \ ll \mapsto \text{arity} x = \text{length} ll) \ t \}$$

- $\text{tree} X$ is the lfp of $\text{tree} X = X \times \text{list} (\text{tree} X)$:

Variable $X : \text{Type}$.

Inductive $\text{tree} : \text{Type} := \text{in\_tree} : X \to \text{list} \ \text{tree} \to \text{tree}$.

- Can freely use the $\text{List}$ library to deal with the forest of sons
- Nested definition does not generate a good elimination scheme
Variable \( P : \text{tree} \to \text{Type} \).

Hypothesis \( f : \forall a \, \forall ll, \left( \forall x, \text{In} x \, \text{ll} \to P x \right) \to P \, (\text{in}_\text{tree} a \, \text{ll}) \).

Definition \( \text{tree}_\text{rect} t : P t := \ldots \) (* use Fix from Wf *)

Hypothesis \( f_\text{ext} : \ldots \)

Fact \( \text{tree}_\text{rect}_\text{fix} a \, \text{ll} : \)
\[
\text{tree}_\text{rect} \left( \text{in}_\text{tree} a \, \text{ll} \right) = f a \, \text{ll} \, (\text{fun} \, t \_ \Rightarrow \text{tree}_\text{rect} t)
\]
Implicit Types (P : X -> list tree -> Prop)
   (Q : nat -> X -> Prop).

Definition tree_fall P : tree -> Prop.

Fact tree_fall_fix P x ll :
   tree_fall P (in_tree x ll)
<-> P x ll
/\ forall t, In t ll -> tree_fall P t.

Let btree k := tree_fall (fun x ll => length ll < k).
Let wfptree Q := tree_fall (fun x ll => Q (length ll) x).
Higman Embedding in Coq

Variables (X : Type) (R : nat -> X -> X -> Prop).

Inductive emb_tree_higman : tree X -> tree X -> Prop :=
  | in_emb_tree_higman_0 : forall s t x ll,
    In t ll
    -> s <eh t
    -> s <eh in_tree x ll
  | in_emb_tree_higman_1 : forall x y ll mm,
    R (length ll) x y
    -> Forall2 emb_tree_higman ll mm
    -> in_tree x ll <eh in_tree y mm

where "x <eh y" := (emb_tree_higman x y).
Variable S : tree X -> tree X -> Prop.
Infix "<<" := S (at level 70).

Hypothesis S_sub0 : forall s t x ll,
    In t ll -> s <eh t
    -> s << t  -> s << in_tree x ll.

Hypothesis S_sub1 : forall x y ll mm, R (length ll) x y
    -> Forall2 emb_tree_higman ll mm
    -> Forall2 S  ll mm
    -> in_tree x ll << in_tree y mm.

Theorem emb_tree_higman_ind t1 t2 : t1 <eh t2 -> t1 << t2.
Almost Full predicate

Definition af_t R := { t : wft X | af_secureby R t }.

Inductive af_type : (X -> X -> Prop) -> Type :=
| in_af_type0 : forall R, full R -> af_type R
| in_af_type1 : forall R, (forall a, af_type (R rlift a)) -> af_type R.

Definition af_t_other R :=
{ t | forall f, good R (pfx_rev f (wft_ht t f)) }.

Thm af_t_eq : af_t R <-> af_type R <-> af_t_other R.
Inductive Bar predicates

Implicit Types (P : list X → Prop) (R : X → X → Prop).

Inductive bar_t P : list X → Type :=
  | in_bar_t0 : ∀ ll, P ll → bar_t P ll
  | in_bar_t1 : ∀ ll, (∀ a, bar_t P (a::ll)) → bar_t P ll.

Inductive good R : list X → Prop :=
  | in_good_0 : ∀ ll a b, In b ll
        → R b a → good R (a::ll)
  | in_good_1 : ∀ ll a, good R ll → good R (a::ll).

Thm af_t_bar_t R : af_t R ↔ bar_t (good R) nil.
A universe tailored for Higman theorem

- Given a type \((X_i)_{i<k}\), a universe \(U\) is a post fixpoint of:

\[
U = \{\ast\} + \sum X_i + U + \mathbb{N} \times U \times \text{tree}U
\]

- Then \(X'_i = X_{i-1} + \sum_{p=0}^{i-1} X_i \times \text{tree}(X_0, \ldots, X_{k-1})\) can be viewed as a sub-type of \(U\) (in Veldman 2004, \(U = \mathbb{N}\))

Variable \(X : \text{Type}\).

Inductive htree_fix :=
| in_htf_u : htree_fix (* undefined *)
| in_htf_0 : X -> htree_fix (* X embeds in U *)
| in_htf_1 : htree_fix -> htree_fix (* U embed in U *)
| in_htf_2 : nat -> htree_fix
  -> tree htree_fix -> htree_fix.
Higman theorem, the recursive statement

Definition owft X := option (wft X).
Variables (X : Type) (k : nat).
Notation U := (htree_fix X).

Theorem higman_htree_rec (s : nat -> owft U) :
  forall P : nat -> U -> Prop,
  (forall n, ~ P n (@in_htf_u X))
-> (forall n x, { P n x } + { ~ P n x })
-> (forall n, k < n -> P n = fun _ => False)
-> forall R,
  (forall n, n <= k -> afs_owft_sec (s n) (P n) (R n))
-> afs_t (wfptree P) (emb_tree_higman R).
What about the logical version

- $\text{af} : \text{rel}_2 X \to \text{Prop}$ instead of $\text{af}_t : \text{rel}_2 X \to \text{Type}$

- Unlike provable/has a proof, $\text{af} \ R$ is NOT inhabited($\text{af}_t \ R$)
  - cannot use empty wft to decide when $R$ is full or not !!

- To get $\text{af} \ R \Rightarrow \exists t, \text{af\_securedby}(t, R)$, you either need:
  - FunctionalChoice_on: $(\forall x \exists y, x R y) \Rightarrow \exists f \forall x, x R f(x)$
  - or Brouwer’s thesis (Veldman 2004)

- How to replace lexicographic induction of wft sequences?
  - first idea: encode lex. product at Prop level instead of Type
  - new idea: use well-founded upto relations
    - wf. upto rels. are stable under lex. products
Variable (X Y : Type).
Implicit Type (f : X -> Y) (R : relation X)
(P : X -> Prop) (Q : Y -> Prop).

Definition well_founded R :=
  forall P, (forall a, (forall b, R b a -> P b)
    -> P a)
  -> forall a, P a.

Definition well_founded_upto f R :=
  forall Q, (forall a, (forall b, R b a -> Q (f b))
    -> Q (f a))
  -> forall a, Q (f a).
Almost full rels and Wf upto 1/2

Inductive afw : Set := af_empty | af_full | af_rlift.
Let lt_afw : afw -> afw -> Prop. (* empty < full < rlift *)

Definition af_subrel := (af_w * ((X -> Prop) * relation X)).

Definition afsr_correct (c : af_subrel) :=
   match c with
     | (af_empty,(P,_)) => forall x, ~ P x
     | (af_full ,(P,R)) => forall x y, P x -> P y -> R x y
     | (af_rlift,(P,R)) => afs P R
   end.
Almost full rels and Wf upto 2/2

Definition lt_afsr (c1 c2 : af_subrel) :=
  match c1 , c2 with
   | (w1,(P1,R1)) , (w2,(P2,R2))
     => lt_afw w1 w2
  \/ w1 = w2
  \/ w1 = af_rlift
  \/ P1 = P2
  \/ exists p, P1 p
  \/ R1 = (R2 rlift p)
end. (* this relation has reflexive elements *)

Theorem lt_afsr_upto_wf :
  well_founded_upto (@snd _ _) afsr_correct lt_afsr.
What about Kruskal’s tree theorem?

- Shares the same structure as Higman theorem
- There are twice as many cases
- The proof uses both Higman lemma and Higman theorem
- The lexicographic product is a bit different: more facile
- The universe is not the same:

\[ U = \{\star\} + X + U + U \times \text{list}\left(\text{list}(\text{tree } U)\right) \]

- Replace \texttt{insert} with the more general \texttt{intercalate}

\[
\text{intercalate } [a_1, \ldots, a_n] [l_0, \ldots, l_n] = l_0 \mathbin{++} a_1 \mathbin{::} \cdots \mathbin{++} a_n \mathbin{::} l_n
\]

- \texttt{emb\_tree\_upto}: inbetween the product and the homeomorphic
Tree Embedding upto $k$

- $\text{tree}(X_n)_{n \in \mathbb{N}} = T$ where $T$ is lfp of $T = \sum_{n=0}^{\infty} X_n \times T^n$

- $k \in \mathbb{N}$ and an arity-indexed relation $R : \forall n \in \mathbb{N}, \text{rel}_2(X_n)$

- one $X_n$ for each arity, but $[X_k = X_n]$ as soon as $n \geq k$

$$s <^u_{k,R} t_i$$

$$\frac{s <^u_{k,R} \langle x_n | t_1, \ldots, t_n \rangle}{s <^u_{k,R} \langle x_n | t_1, \ldots, t_n \rangle}$$

$$n < k \quad x_n \quad R_n \quad y_n \quad s_1 <^u_{k,R} t_1, \ldots, s_n <^u_{k,R} t_n$$

$$\langle x_n | s_1, \ldots, s_n \rangle <^u_{k,R} \langle y_n | t_1, \ldots, t_n \rangle$$

$$k \leq i \quad x_i \quad R_k \quad x_j \quad [s_1, \ldots, s_i] (\text{subword} <^u_{k,R}) [t_1, \ldots, t_j]$$

$$\langle x_i | s_1, \ldots, s_i \rangle <^u_{k,R} \langle x_j | t_1, \ldots, t_j \rangle$$
Coq code for `emb_tree_upto`

Variables (k : nat) (R : nat -> X -> X -> Prop).

Inductive `emb_tree_upto` : tree X -> tree X -> Prop :=

| `in_embut_0` : forall s t x ll, In t ll -> s <eu t
  -> s <eu in_tree x ll |

| `in_embut_1` : forall x y ll mm, length ll < k
  -> R (length ll) x y
  -> Forall2 `emb_tree_upto` ll mm
  -> in_tree x ll <eu in_tree y mm |

| `in_embut_2` : forall x y ll mm, k <= length ll
  -> R k x y
  -> subword `emb_tree_upto` ll mm
  -> in_tree x ll <eu in_tree y mm |

where "x <eu y" := (emb_tree_upto x y).
Kruskal’s Tree Theorem, the recursive statement

Variables (X : Type).
Notation U := (ktree_fix X).

Theorem kruskal_ktree_rec (s : nt_stump U) :
  forall k, k = nts_char s
  ->forall (P : nat -> U -> Prop),
     (forall n, ~ P n in_ktf_u)
     -> (forall n x, { P n x } + { ~ P n x })
     -> (forall n, k <= n -> P k = P n)
  ->forall (R : nat -> relation U),
     (forall n, n <= k
      -> afs_owft_sec (nts_seq s n) (P n) (R n))
  -> afs_t (wfptree P) (emb_tree_uppto k R).