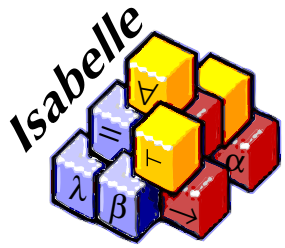
A constructive proof of Higman's lemma in Isabelle

Stefan Berghofer
Institut für Informatik, TU München

joint work with

Monika Seisenberger
Department of Computer Science, University of Wales Swansea



Background

Higman's lemma

Every infinite sequence of words $(w_i)_{0 \leq i < \omega}$ contains two words w_i and w_j with $i < j$ such that w_i can be embedded into w_j (denoted by $w_i \trianglelefteq w_j$).

A sequence with the above property is sometimes called good, otherwise bad.

- Interesting problem from the field of combinatorics
- Specific instance of Kruskal's tree theorem
- **Application:** proving termination of term rewriting systems using simplification orders (i.e. orders $>$ on terms $T(\Sigma, V)$, which are compatible with Σ-operations, closed under substitutions, and have the subterm property $\forall p \in \mathcal{P}os(t) - \{\epsilon\}.\ t > t|_p$)

Why a constructive proof?

- Additional informative content inherent in constructive proofs
- Can e.g. be used to obtain upper bounds on length of reduction sequences from termination proof of string rewrite system
- Possibility to automatically derive an algorithm via program extraction

History

- 1952: Higman's original paper
- Nash-Williams 1963: Short and elegant classical proof using minimal bad sequence argument
- Murthy 1990: First formalization using a theorem prover (Nuprl)
  Based on Nash-Williams' classical proof, turned into constructive proof via double negation / A translation $\Longrightarrow$ Extracted program about 12 MB in size :-(
- Coquand and Fridlender 1993: Elegant and short constructive proof, based entirely on inductive definitions
- Herbelin 1994: Formalization of translated classical proof in Coq (impredicative)
- Fridlender 1996: Formalization using ALF proof editor, based on paper proof by Veldman
- Seisenberger 1998: Formalization of Coquand's and Fridlender's proof in MINLOG

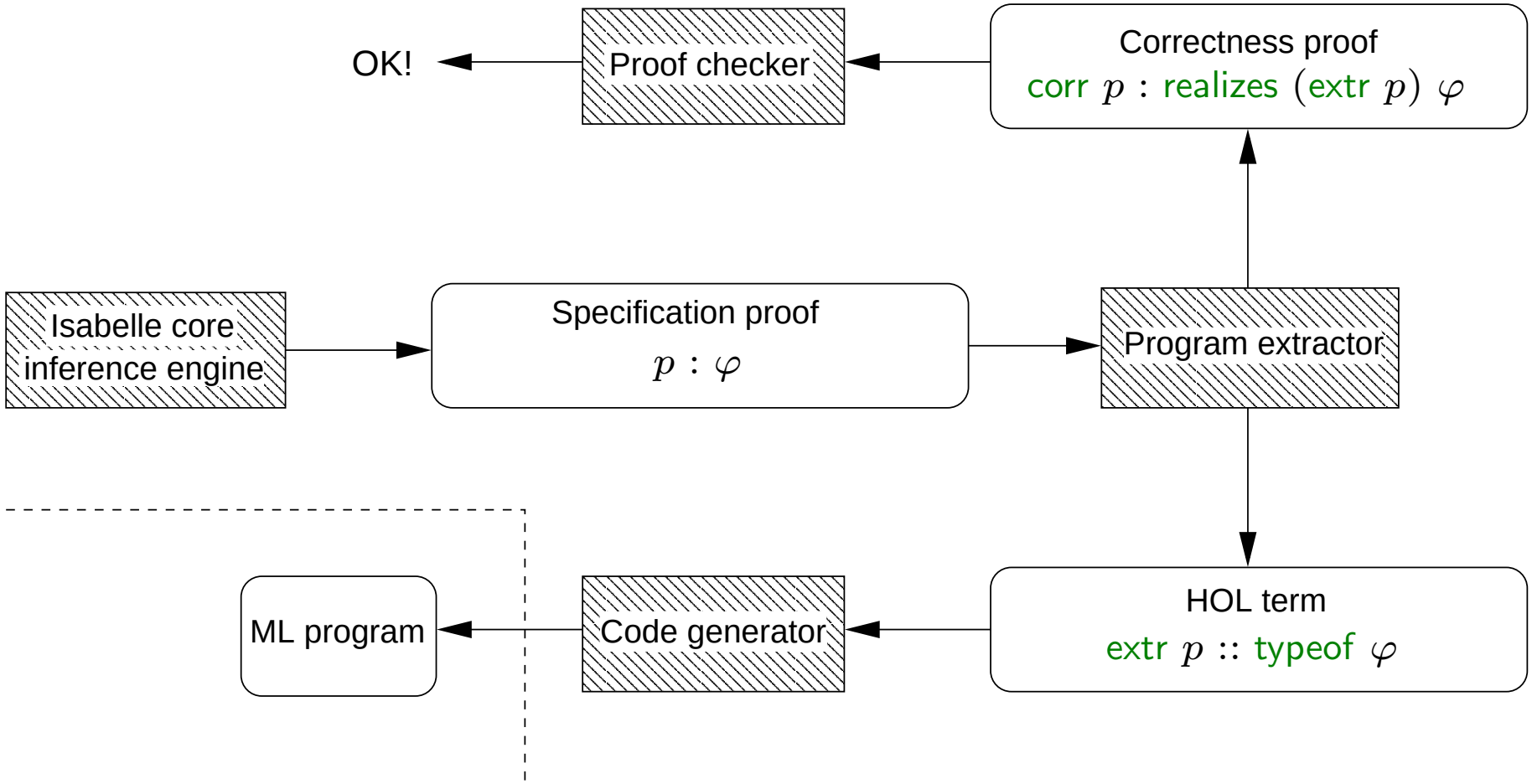
What's new?

- Extraction framework automatically produces correctness proof, checkable inside the system (not just on paper)
- Readable proofs using Isabelle/Isar language (by M. Wenzel)
- Intuitive graphical description of computational behaviour
- Modular program

The rest of the talk ...

- Program extraction in Isabelle
- Basic definitions for Higman's lemma
- Classical proof
- Constructive proof & extracted programs
- Conclusion

A framework for program extraction



Higman's lemma — Basic definitions

Words

Modelled as lists from the two letter alphabet

datatype $letter = A \mid B$

The embedding relation on words

consts $emb :: (letter\ list \times letter\ list)\ set$
inductive emb
intros
$emb0$: $[] \trianglelefteq bs$
$emb1$: $as \trianglelefteq bs \Longrightarrow as \trianglelefteq b \# bs$
$emb2$: $as \trianglelefteq bs \Longrightarrow a \# as \trianglelefteq a \# bs$

Intuition: as can be embedded into bs, if we can obtain as by deleting letters from bs.
Example: $[A, A] \trianglelefteq [B, A, B, A]$

Good sequences

Set of words containing a word which can be embedded into y

consts $L :: letter\ list \Rightarrow letter\ list\ list\ set$
inductive $L\ y$
intros
$L0\colon w \trianglelefteq y \Longrightarrow w \# ws \in L\ y$
$L1\colon ws \in L\ y \Longrightarrow w \# ws \in L\ y$

consts $good :: letter\ list\ list\ set$
inductive $good$
intros
$good0\colon ws \in L\ w \Longrightarrow w \# ws \in good$
$good1\colon ws \in good \Longrightarrow w \# ws \in good$

Caveat: $good$ works in "opposite" direction, since words are appended to the left!
Example: $[[A,\ A],\ [A,\ B],\ [B]] \in good$, since $[B] \trianglelefteq [A,\ B]$

The bar predicate

consts $bar :: letter\ list\ list\ set$
inductive bar
intros
$bar1$: $ws \in good \Longrightarrow ws \in bar$
$bar2$: $(\bigwedge w.\ w\ \#\ ws \in bar) \Longrightarrow ws \in bar$

Intuition: $ws \in bar \quad \approx$ Either ws is already $good$, or can be turned into a good sequence by adding words.

Consequence: $[] \in bar \quad \approx$ Every infinite sequence $(w_i)_{0 \leq i < \omega}$ must be good, i.e. have a prefix $w_0\ \ldots\ w_n$ with $[w_n,\ \ldots,\ w_0] \in good$.

Closely related to Brouwer's more general principle of bar induction

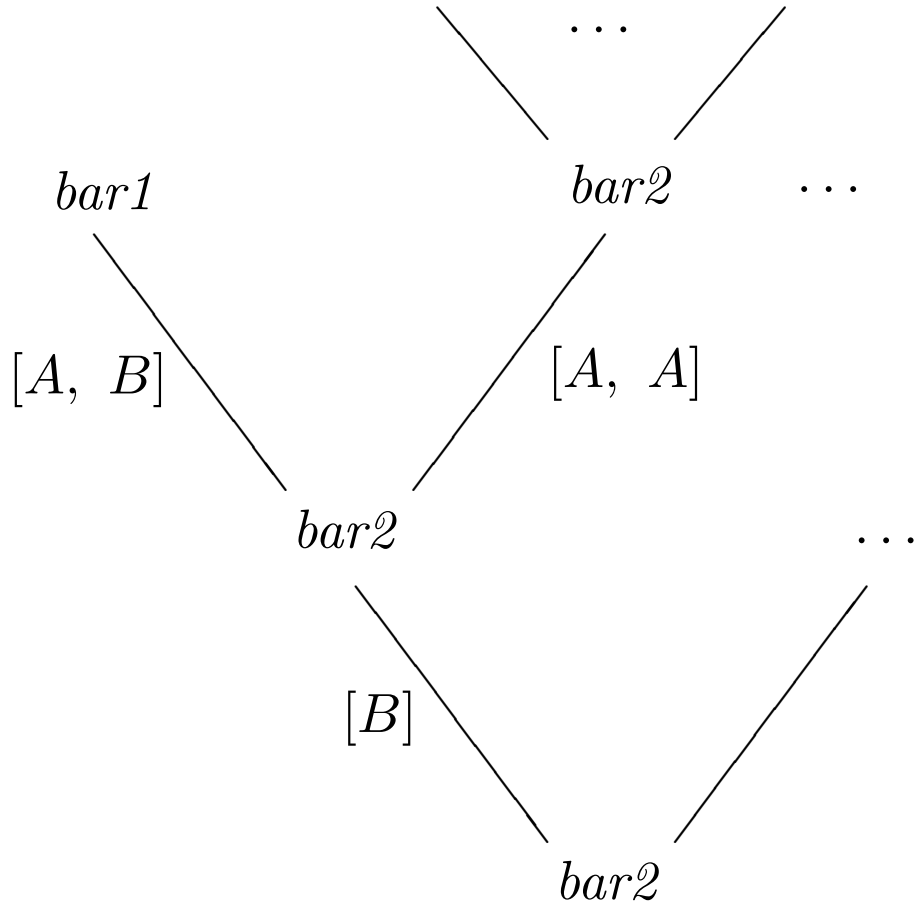
Computational content of inductive datatypes and predicates

Proof	Program
induction on datatype $nat\text{-}induct : P\ 0 \Longrightarrow$ $(\bigwedge x.\ P\ x \Longrightarrow P\ (Suc\ x)) \Longrightarrow P\ z$	recursion on datatype $nat\text{-}rec : nat \Rightarrow \alpha_P \Rightarrow$ $(nat \Rightarrow \alpha_P \Rightarrow \alpha_P) \Rightarrow \alpha_P$
inductive predicate introduction rules **inductive** bar $bar1 : \bigwedge ws.\ ws \in good \Longrightarrow ws \in bar$ $bar2 : \bigwedge ws.\ (\bigwedge w.\ w \# ws \in bar) \Longrightarrow$ $ws \in bar$	inductive datatype constructors **datatype** $barT =$ $bar1\ (letter\ list\ list)$ $\mid bar2\ (letter\ list\ list)\ (letter\ list \Rightarrow barT)$
induction on derivation $bar\text{-}induct : vs \in bar \Longrightarrow$ $(\bigwedge ws.\ ws \in good \Longrightarrow P\ ws) \Longrightarrow$ $(\bigwedge ws.\ (\bigwedge w.\ w \# ws \in bar) \Longrightarrow$ $(\bigwedge w.\ P\ (w \# ws)) \Longrightarrow P\ ws) \Longrightarrow$ $P\ vs$	recursion on datatype $bar\text{-}rec : barT \Rightarrow$ $(letter\ list\ list \Rightarrow \alpha_P) \Rightarrow$ $(letter\ list\ list \Rightarrow (letter\ list \Rightarrow barT) \Rightarrow$ $(letter\ list \Rightarrow \alpha_P) \Rightarrow \alpha_P) \Rightarrow$ α_P

Computational content of bar predicate

datatype $barT \;=\; bar1\ (letter\ list\ list) \;\mid\; bar2\ (letter\ list\ list)\ (letter\ list \Rightarrow barT)$

Infinitely branching, well-founded (search) tree:



Realizability for bar predicate

Note: realizes $b\ (ws \in bar) = (b,\ ws) \in barR$

Intuition: $(b,\ ws) \in barR \quad \approx \quad b$ is a correct witness for $ws \in bar$

Inductive characterization of $barR$

$$ws \in good \Longrightarrow (bar1\ ws,\ ws) \in barR$$
$$(\bigwedge w.\ (f\ w,\ w\ \#\ ws) \in barR) \Longrightarrow (bar2\ ws\ f,\ ws) \in barR$$

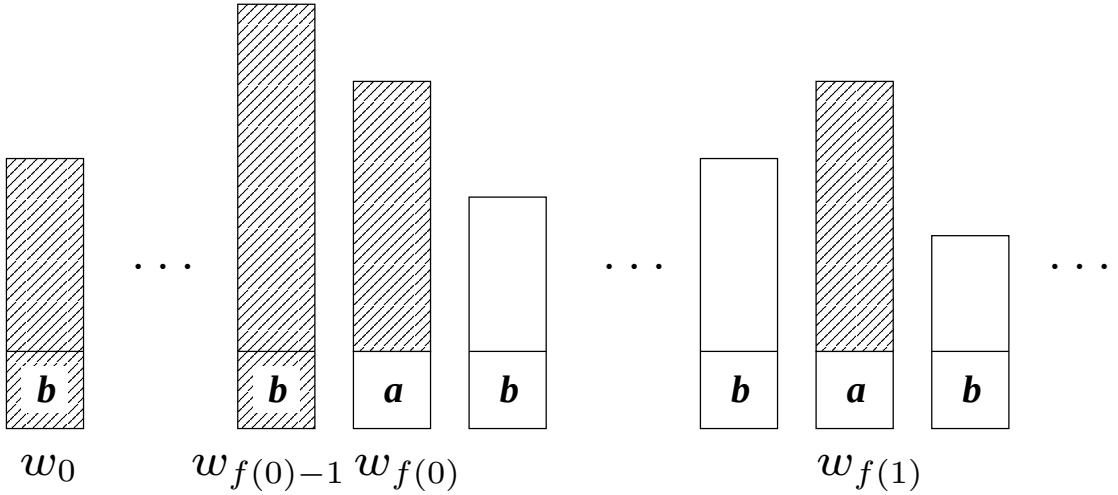
Consequence: If $(bar2\ ws\ f_0,\ ws) \in barR$ and

$$f_0\ w_0 = bar2\ (w_0\ \#\ ws)\ f_1, \quad f_1\ w_1 = bar2\ (w_1\ \#\ w_0\ \#\ ws)\ f_2, \quad \ldots,$$
$$f_{n-1}\ w_{n-1} = bar2\ (w_{n-1}\ \#\ \cdots\ \#\ w_0\ \#\ ws)\ f_n, \quad f_n\ w_n = bar1\ (w_n\ \#\ \cdots\ \#\ w_0\ \#\ ws)$$

then $(w_n\ \#\ \cdots\ \#\ w_0\ \#\ ws) \in good$.

Note: This is not necessarily the shortest possible $good$ sequence.

Nash-Williams' minimal bad sequence argument



Proof by contradiction: Assume there is a bad sequence. Can construct bad sequence $(w_i)_{1\leq i<\omega}$ which is minimal wrt. word length. Note that $w_i = a_i \# v_i$. Can find strictly monotone f and $a \in \{A, B\}$ such that $a_{f(i)} = a$ for all i. Now consider the sequence $(v_{f(i)})_{0\leq i<\omega}$. If this sequence was bad, could construct sequence

$$s \;=\; w_0 \;\ldots\; w_{f(0)-1}\; v_{f(0)}\; v_{f(1)} \;\ldots$$

Because $len(v_{f(0)}) < len(w_{f(0)})$, and $(w_i)_{1\leq i<\omega}$ minimal, this sequence must be good. Hence must have i, j with $i < f(0)$, $f(0) \leq j$ and $w_i \trianglelefteq v_{f(j)}$. Since $v_{f(j)} \trianglelefteq w_{f(j)}$, this implies $w_i \trianglelefteq w_{f(j)}$. Contradiction! Hence $(v_{f(i)})_{0\leq i<\omega}$ good, i.e. there are i, j with $i < j$ and $v_{f(i)} \trianglelefteq v_{f(j)}$, which implies $a \# v_{f(i)} \trianglelefteq a \# v_{f(j)}$ and $w_{f(i)} \trianglelefteq w_{f(j)}$. Contradiction!

A constructive version of the minimal bad sequence argument

Prefixing words with a letter

consts $R :: letter \Rightarrow (letter\ list\ list \times letter\ list\ list)\ set$
inductive $R\ a$
intros
$R0$: $([], []) \in R\ a$
$R1$: $(vs, ws) \in R\ a \Longrightarrow (w \# vs, (a \# w) \# ws) \in R\ a$

Constructing the sequence $s = w_0 \ldots w_{f(0)-1}\ v_{f(0)}\ v_{f(1)} \ldots$

consts $T :: letter \Rightarrow (letter\ list\ list \times letter\ list\ list)\ set$
inductive $T\ a$
intros
$T0$: $a \neq b \Longrightarrow (vs, ws) \in R\ b \Longrightarrow (w \# ws, (a \# w) \# ws) \in T\ a$
$T1$: $(vs, ws) \in T\ a \Longrightarrow (v \# vs, (a \# v) \# ws) \in T\ a$
$T2$: $a \neq b \Longrightarrow (vs, ws) \in T\ a \Longrightarrow (vs, (b \# w) \# ws) \in T\ a$

Example:

$([w_5, w_2, B \# w_1, B \# w_0],$
$[B \# w_6, A \# w_5, B \# w_4, B \# w_3, A \# w_2, B \# w_1, B \# w_0]) \in T\ A$

Proof roadmap

Computational view: Functions transforming trees

prop1: **Sequences "ending" with empty word** (trivial)

$$([] \# ws) \in bar$$

prop2: **Interleaving two trees**

$$a \neq b \Longrightarrow xs \in bar \Longrightarrow ys \in bar \Longrightarrow (xs, zs) \in T\ a \Longrightarrow (ys, zs) \in T\ b \Longrightarrow zs \in bar$$

prop3: **Lifting to longer words**

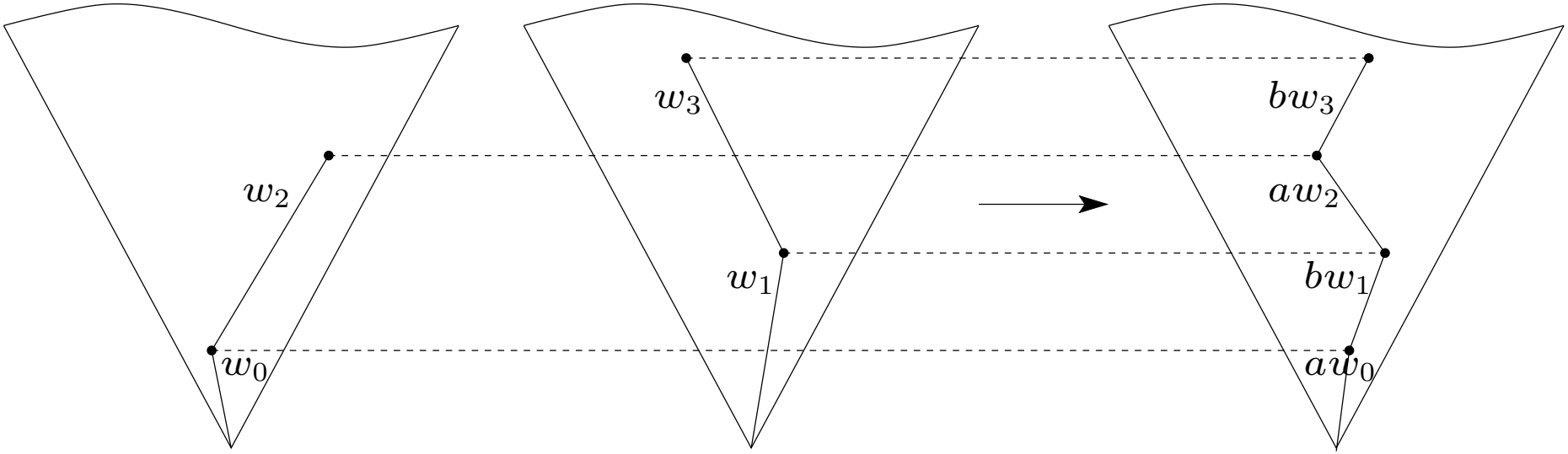
$$xs \in bar \Longrightarrow xs \neq [] \Longrightarrow (xs, zs) \in R\ a \Longrightarrow zs \in bar$$

higman: **Main theorem**

$$[] \in bar$$

Interleaving two trees — *prop2*

$$a \neq b \Longrightarrow xs \in bar \Longrightarrow ys \in bar \Longrightarrow (xs,\ zs) \in T\ a \Longrightarrow (ys,\ zs) \in T\ b \Longrightarrow zs \in bar$$



Proof idea: Induction on $xs \in bar$ and $ys \in bar$, then case distinction on first letter of first word following zs

Isabelle/Isar proof of $prop2$

theorem $prop2$:
  assumes ab: $a \neq b$ **and** bar: $xs \in bar$
  shows $\bigwedge ys\ zs.\ ys \in bar \Longrightarrow (xs,\ zs) \in T\ a \Longrightarrow (ys,\ zs) \in T\ b \Longrightarrow zs \in bar$ **using** bar
proof $induct$
  fix $xs\ zs$ **assume** $xs \in good$ **and** $(xs,\ zs) \in T\ a$
  show $zs \in bar$ **by** $(rule\ bar1)\ (rule\ lemma3)$
next
  fix $xs\ ys$
  assume I: $\bigwedge w\ ys\ zs.\ ys \in bar \Longrightarrow (w\ \#\ xs,\ zs) \in T\ a \Longrightarrow (ys,\ zs) \in T\ b \Longrightarrow zs \in bar$
  assume $ys \in bar$
  thus $\bigwedge zs.\ (xs,\ zs) \in T\ a \Longrightarrow (ys,\ zs) \in T\ b \Longrightarrow zs \in bar$
  proof $induct$
    fix $ys\ zs$ **assume** $ys \in good$ **and** $(ys,\ zs) \in T\ b$
    show $zs \in bar$ **by** $(rule\ bar1)\ (rule\ lemma3)$
  next
    fix $ys\ zs$ **assume** I': $\bigwedge w\ zs.\ (xs,\ zs) \in T\ a \Longrightarrow (w\ \#\ ys,\ zs) \in T\ b \Longrightarrow zs \in bar$
    and ys: $\bigwedge w.\ w\ \#\ ys \in bar$ **and** Ta: $(xs,\ zs) \in T\ a$ **and** Tb: $(ys,\ zs) \in T\ b$
    show $zs \in bar$

Isabelle/Isar proof of $prop2$ — continued

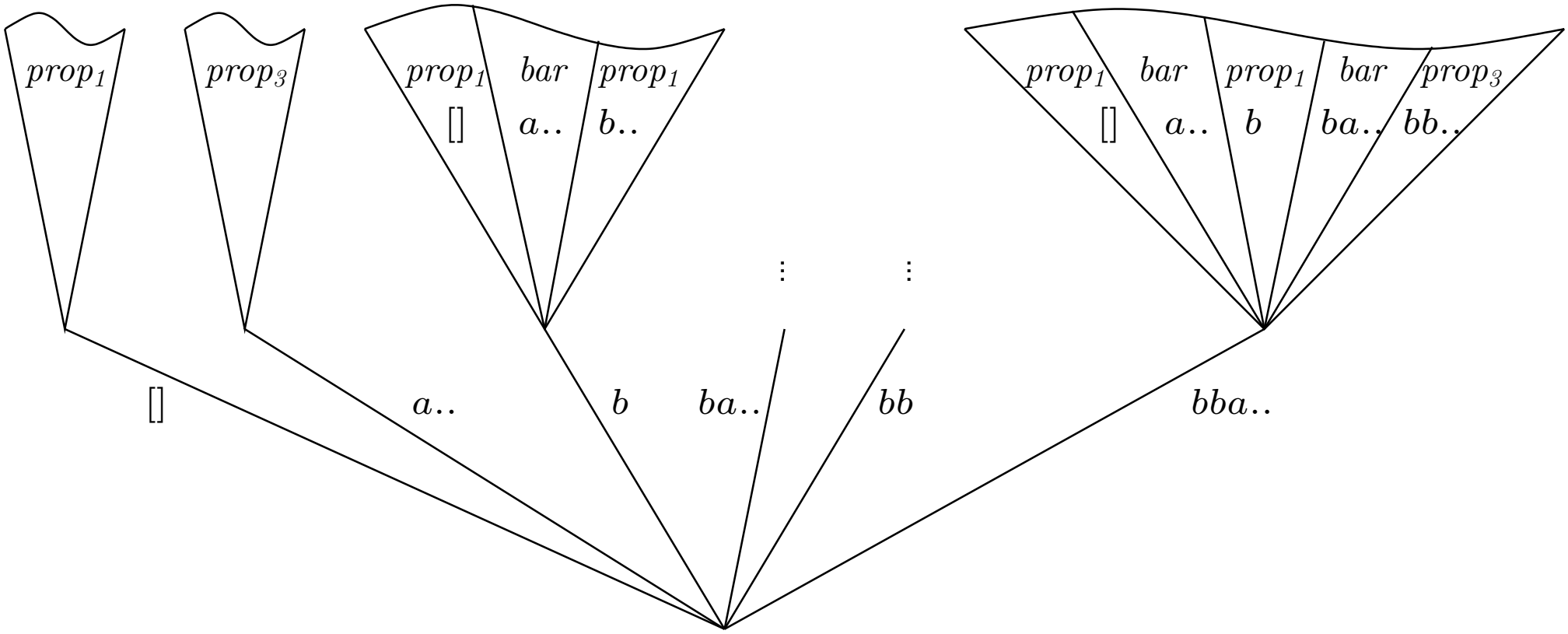
proof $(rule\ bar2)$
  fix w **show** $w\ \#\ zs \in bar$
  proof $(cases\ w)$
    case Nil **thus** $?thesis$ **by** $simp\ (rule\ prop1)$
  next
    case $(Cons\ c\ cs)$ **from** $letter\text{-}eq\text{-}dec$ **show** $?thesis$
    proof
      assume ca: $c = a$
      from ab **have** $(a\ \#\ cs)\ \#\ zs \in bar$ **by** $(rules\ intro{:}\ I\ ys\ Ta\ Tb)+$
      thus $?thesis$ **by** $(simp\ add{:}\ Cons\ ca)$
    next
      assume $c \neq a$ **with** ab **have** cb: $c = b$ **by** $(rule\ letter\text{-}neq)$
      from ab **have** $(b\ \#\ cs)\ \#\ zs \in bar$ **by** $(rules\ intro{:}\ I'\ Ta\ Tb)+$
      thus $?thesis$ **by** $(simp\ add{:}\ Cons\ cb)$
    qed
  qed
qed
qed

Program extracted from *prop2*

$$
\begin{array}{l}
prop2 \equiv \\
\lambda x\ xa\ xb\ xc\ H\ Ha. \\
\quad barT\text{-}rec\ (\lambda ws\ x\ xa\ H.\ bar1\ xa) \\
\quad\ (\lambda ws\ xb\ r\ xc\ xd\ H. \\
\qquad barT\text{-}rec\ (\lambda ws\ x.\ bar1\ x) \\
\qquad\ (\lambda ws\ xb\ ra\ xc. \\
\qquad\quad bar2\ xc \\
\qquad\quad\ (\lambda w.\ case\ w\ of\ [] \Rightarrow prop1\ xc \\
\qquad\qquad\qquad \mid a\ \#\ list \Rightarrow \\
\qquad\qquad\qquad\quad case\ letter\text{-}eq\text{-}dec\ a\ x\ of\ Left \Rightarrow r\ list\ ws\ ((x\ \#\ list)\ \#\ xc)\ (bar2\ ws\ xb) \\
\qquad\qquad\qquad\quad \mid Right \Rightarrow ra\ list\ ((xa\ \#\ list)\ \#\ xc))) \\
\qquad H\ xd) \\
\quad\ H\ xb\ xc\ Ha
\end{array}
$$

Lifting to longer words — *prop3*

$$xs \in bar \Longrightarrow xs \neq [] \Longrightarrow (xs,\ zs) \in R\ a \Longrightarrow zs \in bar$$



Proof idea: Induction on $xs \in bar$, then induction on first word following zs

Isabelle/Isar proof of $prop3$

theorem $prop3$:
  assumes bar: $xs \in bar$
  shows $\bigwedge zs.\ xs \neq [] \Longrightarrow (xs,\ zs) \in R\ a \Longrightarrow zs \in bar$ **using** bar
proof $induct$
  fix $xs\ zs$
  assume $xs \in good$ **and** $(xs,\ zs) \in R\ a$
  show $zs \in bar$ **by** $(rule\ bar1)\ (rule\ lemma2)$
next
  fix $xs\ zs$
  assume I: $\bigwedge w\ zs.\ w\ \#\ xs \neq [] \Longrightarrow (w\ \#\ xs,\ zs) \in R\ a \Longrightarrow zs \in bar$
  and xsb: $\bigwedge w.\ w\ \#\ xs \in bar$ **and** xsn: $xs \neq []$ **and** R: $(xs,\ zs) \in R\ a$
  show $zs \in bar$
  proof $(rule\ bar2)$
    fix w
    show $w\ \#\ zs \in bar$
    proof $(induct\ w)$
      case Nil
      show $?case$ **by** $(rule\ prop1)$
    next

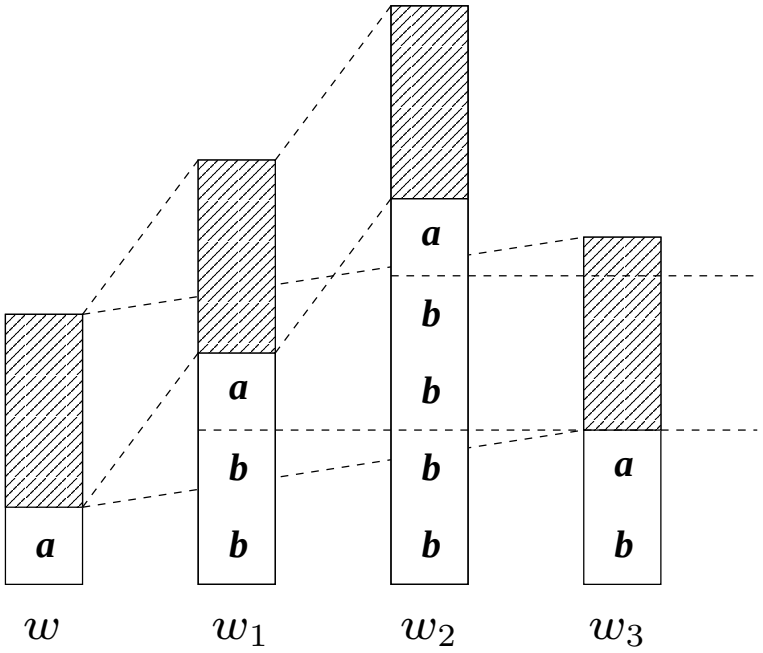
Isabelle/Isar proof of *prop3* — continued

case (*Cons c cs*)
from *letter-eq-dec* **show** *?case*
proof
assume $c = a$
thus *?thesis* **by** (*rules intro*: *I* [*simplified*] *R*)
next
from *R xsn* **have** *T*: (*xs*, *zs*) ∈ *T a* **by** (*rule lemma4*)
assume $c \neq a$
thus *?thesis* **by** (*rules intro*: *prop2 Cons xsb xsn R T*)
qed
qed
qed
qed

Program extracted from *prop3*

$$
\begin{array}{l}
prop3 \equiv \\
\lambda x\ xa\ H. \\
\quad barT\text{-}rec\ (\lambda ws.\ bar1) \\
\quad (\lambda ws\ x\ r\ xb. \\
\qquad bar2\ xb \\
\qquad (list\text{-}rec\ (prop1\ xb) \\
\qquad\ (\lambda a\ list\ H. \\
\qquad\qquad case\ letter\text{-}eq\text{-}dec\ a\ xa\ of\ Left \Rightarrow r\ list\ ((xa \#\ list) \#\ xb) \\
\qquad\qquad \mid\ Right \Rightarrow prop2\ a\ xa\ ws\ ((a \#\ list) \#\ xb)\ H\ (bar2\ ws\ x)))) \\
\quad H\ x
\end{array}
$$

The main theorem



theorem *higman*: $[] \in bar$
proof (*rule bar2*)
  fix w
  show $[w] \in bar$
  proof (*induct w*)
    show $[[]] \in bar$ **by** (*rule prop1*)
  next
    fix $c\ cs$ **assume** $[cs] \in bar$
    thus $[c \# cs] \in bar$
      by (*rule prop3*) (*simp, rules*)
  qed
qed

Extracted program

$higman \equiv bar2\ []\ (list\text{-}rec\ (prop1\ [])\ (\lambda a\ list.\ prop3\ [a \# list]\ a))$

How to test this program?

Finding good prefixes of infinite sequences

consts $is\text{-}prefix :: 'a\ list \Rightarrow (nat \Rightarrow 'a) \Rightarrow bool$
primrec
$is\text{-}prefix\ []\ f = True$
$is\text{-}prefix\ (x\ \#\ xs)\ f = (x = f\ (length\ xs) \wedge is\text{-}prefix\ xs\ f)$

theorem $good\text{-}prefix\text{-}lemma$:
assumes bar: $ws \in bar$
shows $is\text{-}prefix\ ws\ f \Longrightarrow \exists\, vs.\ is\text{-}prefix\ vs\ f \wedge vs \in good$ **using** bar
proof $induct$
case $bar1$
thus $?case$ **by** $rules$
next
case $(bar2\ ws)$
have $is\text{-}prefix\ (f\ (length\ ws)\ \#\ ws)\ f$ **by** $simp$
thus $?case$ **by** $(rules\ intro: bar2)$
qed

theorem $good\text{-}prefix$: $\exists\, vs.\ is\text{-}prefix\ vs\ f \wedge vs \in good$ **using** $higman$
by $(rule\ good\text{-}prefix\text{-}lemma)\ simp+$

Program extracted from *good-prefix*

$$good\text{-}prefix\text{-}lemma \equiv \lambda x.\ barT\text{-}rec\ (\lambda ws.\ ws)\ (\lambda ws\ xa\ r.\ r\ (x\ (length\ ws)))$$

$$good\text{-}prefix \equiv \lambda x.\ good\text{-}prefix\text{-}lemma\ x\ higman$$

Correctness theorems

Correctness of *good-prefix*

$$is\text{-}prefix\ (good\text{-}prefix\ f)\ f \wedge good\text{-}prefix\ f \in good$$

Correctness of *higman*

$$(higman,\ []) \in barR$$

Correctness of *prop2*

$$\begin{array}{l} a \neq b \Longrightarrow \\ (\bigwedge x.\ (x,\ xs) \in barR \Longrightarrow \\ \quad (\bigwedge xa.\ (xa,\ ys) \in barR \Longrightarrow \\ \quad\quad (xs,\ zs) \in T\ a \Longrightarrow (ys,\ zs) \in T\ b \Longrightarrow (prop2\ a\ b\ ys\ zs\ x\ xa,\ zs) \in barR)) \end{array}$$

Conclusion

- Realistic example for program extraction
- Automatically extracted program quite readable ($\approx$ 70 lines of ML code, including auxiliary functions)
- Compact formalization ($\approx$ 280 lines of Isabelle code)
- Performs reasonably well on medium-size sequences (e.g. 0.8 sec for sequence of 100 words with length $>$ 15, on Pentium III with 700 MHz)
- TODO:
  - Extend to arbitrary finite alphabet (already done in MINLOG by Monika Seisenberger)
  - Complexity analysis

Interested in more examples? $\Longrightarrow$ See the demo!