Formalized Soundness and Completeness of Natural Deduction for First-Order Logic

Local Isabelle Workshop 6/6 2018

Andreas Halkjær From

DTU Compute
Department of Applied Mathematics and Computer Science

Introduction

Several benefits to formalizing proofs:

- Less room for error (if any).
- No parts left as *exercise for the reader*.
- Proof can be explored interactively.
- It's fun!

At DTU we are interested in natural deduction for teaching purposes (NaDeA).

Abstract referred to my extension of Berghofer's work.
For continuity with previous talk I will use NaDeA here.

Agenda

- Isabelle & NaDeA
- Soundness
- Closed Formulas
- Open Formulas
- Conclusion

Syntax

LCF-style theorem prover based on Higher-Order Logic (HOL).

All proofs go through (small) kernel of axioms and inference rules.

Like functional programming with datatypes for First-Order Logic.

Proofs are written in the declarative language Isar.

Syntax

LCF-style theorem prover based on Higher-Order Logic (HOL).
All proofs go through (small) kernel of axioms and inference rules.
Like functional programming with datatypes for First-Order Logic.
Proofs are written in the declarative language Isar.

Terms

type-synonym *id* = *char list*

datatype *tm* = *Var nat* | *Fun id tm list*

Formulas

datatype *fm* = *Falsity* | *Pre id tm list* | *Imp fm fm* | *Dis fm fm* | *Con fm fm* | *Exi fm* | *Uni fm*

Semantics I

Type variable $'a$ encodes domain.
Environment $e :: nat \Rightarrow {'a}$.
Function denotation: $f :: id \Rightarrow {'a}\ list \Rightarrow {'a}$.

Terms

primrec

$$
\begin{aligned}
&\mathit{semantics\text{-}term} :: (nat \Rightarrow {'a}) \Rightarrow (id \Rightarrow {'a}\ list \Rightarrow {'a}) \Rightarrow tm \Rightarrow {'a}\ \textbf{and} \\
&\mathit{semantics\text{-}list} :: (nat \Rightarrow {'a}) \Rightarrow (id \Rightarrow {'a}\ list \Rightarrow {'a}) \Rightarrow tm\ list \Rightarrow {'a}\ list\ \textbf{where} \\
&\mathit{semantics\text{-}term}\ e\ f\ (Var\ n) = \boxed{e\ n}\ | \\
&\mathit{semantics\text{-}term}\ e\ f\ (Fun\ i\ l) = \boxed{f\ i\ (\mathit{semantics\text{-}list}\ e\ f\ l)}\ | \\
&\mathit{semantics\text{-}list}\ e\ f\ [] = []\ | \\
&\mathit{semantics\text{-}list}\ e\ f\ (t \# l) = \mathit{semantics\text{-}term}\ e\ f\ t \# \mathit{semantics\text{-}list}\ e\ f\ l
\end{aligned}
$$

Semantics II

Predicate denotation: $g :: id \Rightarrow {}'a\ list \Rightarrow bool$.

Formulas

primrec

$semantics :: (nat \Rightarrow {}'a) \Rightarrow (id \Rightarrow {}'a\ list \Rightarrow {}'a) \Rightarrow (id \Rightarrow {}'a\ list \Rightarrow bool) \Rightarrow fm \Rightarrow bool$

where

$semantics\ e\ f\ g\ Falsity = False\ |$

$semantics\ e\ f\ g\ (Pre\ i\ l) = g\ i\ (semantics\text{-}list\ e\ f\ l)\ |$

$semantics\ e\ f\ g\ (Imp\ p\ q) = (if\ semantics\ e\ f\ g\ p\ then\ semantics\ e\ f\ g\ q\ else\ True)\ |$

$semantics\ e\ f\ g\ (Dis\ p\ q) = (if\ semantics\ e\ f\ g\ p\ then\ True\ else\ semantics\ e\ f\ g\ q)\ |$

$semantics\ e\ f\ g\ (Con\ p\ q) = (if\ semantics\ e\ f\ g\ p\ then\ semantics\ e\ f\ g\ q\ else\ False)\ |$

$semantics\ e\ f\ g\ (Exi\ p) = (\exists x.\ semantics\ (\lambda n.\ if\ n = 0\ then\ x\ else\ e\ (n - 1))\ f\ g\ p)\ |$

$semantics\ e\ f\ g\ (Uni\ p) = (\forall x.\ semantics\ (\lambda n.\ if\ n = 0\ then\ x\ else\ e\ (n - 1))\ f\ g\ p)$

From rule to code

$$\frac{\phi \in \Gamma}{\Gamma \vdash \phi}\ \textit{assum}$$

From rule to code

$$\frac{\phi \in \Gamma}{\Gamma \vdash \phi}\ \textit{assum}$$

$$\frac{p \in z}{z \vdash p}\ \textit{assum}$$

From rule to code

$$\frac{\phi \in \Gamma}{\Gamma \vdash \phi}\ \textit{assum}$$

$$\frac{p \in z}{z \vdash p}\ \textit{assum}$$

$$\frac{\textit{member p z}}{\textit{OK p z}}\ \textit{Assume}$$

From rule to code

$$\frac{\phi \in \Gamma}{\Gamma \vdash \phi}\ \textit{assum}$$

$$\frac{p \in z}{z \vdash p}\ \textit{assum}$$

$$\frac{\textit{member p z}}{\textit{OK p z}}\ \textit{Assume}$$

$$\textit{Assume}\text{: } \textit{member p z} \Longrightarrow \textit{OK p z}$$

OK

inductive $\mathit{OK} :: \mathit{fm} \Rightarrow \mathit{fm\ list} \Rightarrow \mathit{bool}$ **where**

Assume: $\mathit{member}\ p\ z \Longrightarrow \mathit{OK}\ p\ z \mid$
Boole: $\mathit{OK}\ \mathit{Falsity}\ ((\mathit{Imp}\ p\ \mathit{Falsity})\ \#\ z) \Longrightarrow \mathit{OK}\ p\ z \mid$
$\mathit{Imp\text{-}E}$: $\mathit{OK}\ (\mathit{Imp}\ p\ q)\ z \Longrightarrow \mathit{OK}\ p\ z \Longrightarrow \mathit{OK}\ q\ z \mid$
$\mathit{Imp\text{-}I}$: $\mathit{OK}\ q\ (p\ \#\ z) \Longrightarrow \mathit{OK}\ (\mathit{Imp}\ p\ q)\ z \mid$
$\mathit{Dis\text{-}E}$: $\mathit{OK}\ (\mathit{Dis}\ p\ q)\ z \Longrightarrow \mathit{OK}\ r\ (p\ \#\ z) \Longrightarrow \mathit{OK}\ r\ (q\ \#\ z) \Longrightarrow \mathit{OK}\ r\ z \mid$
$\mathit{Dis\text{-}I1}$: $\mathit{OK}\ p\ z \Longrightarrow \mathit{OK}\ (\mathit{Dis}\ p\ q)\ z \mid$
$\mathit{Dis\text{-}I2}$: $\mathit{OK}\ q\ z \Longrightarrow \mathit{OK}\ (\mathit{Dis}\ p\ q)\ z \mid$
$\mathit{Con\text{-}E1}$: $\mathit{OK}\ (\mathit{Con}\ p\ q)\ z \Longrightarrow \mathit{OK}\ p\ z \mid$
$\mathit{Con\text{-}E2}$: $\mathit{OK}\ (\mathit{Con}\ p\ q)\ z \Longrightarrow \mathit{OK}\ q\ z \mid$
$\mathit{Con\text{-}I}$: $\mathit{OK}\ p\ z \Longrightarrow \mathit{OK}\ q\ z \Longrightarrow \mathit{OK}\ (\mathit{Con}\ p\ q)\ z \mid$
$\mathit{Exi\text{-}E}$: $\mathit{OK}\ (\mathit{Exi}\ p)\ z \Longrightarrow \mathit{OK}\ q\ ((\mathit{sub}\ 0\ (\mathit{Fun}\ c\ [])\ p)\ \#\ z) \Longrightarrow$
$\qquad\qquad \mathit{news}\ c\ (p\ \#\ q\ \#\ z) \Longrightarrow \mathit{OK}\ q\ z \mid$
$\mathit{Exi\text{-}I}$: $\mathit{OK}\ (\mathit{sub}\ 0\ t\ p)\ z \Longrightarrow \mathit{OK}\ (\mathit{Exi}\ p)\ z \mid$
$\mathit{Uni\text{-}E}$: $\mathit{OK}\ (\mathit{Uni}\ p)\ z \Longrightarrow \mathit{OK}\ (\mathit{sub}\ 0\ t\ p)\ z \mid$
$\mathit{Uni\text{-}I}$: $\mathit{OK}\ (\mathit{sub}\ 0\ (\mathit{Fun}\ c\ [])\ p)\ z \Longrightarrow \mathit{news}\ c\ (p\ \#\ z) \Longrightarrow \mathit{OK}\ (\mathit{Uni}\ p)\ z$

Soundness

Context

lemma *soundness'*:
  $OK\ p\ z \Longrightarrow \textit{list-all}\ (\textit{semantics}\ e\ f\ g)\ z \Longrightarrow \textit{semantics}\ e\ f\ g\ p$

Proof by induction over inference rules. Written declaratively:

case $(\textit{Uni-I}\ c\ p\ z)$
then have $\forall x.\ \textit{list-all}\ (\textit{semantics}\ e\ (f(c := \lambda w.\ x))\ g)\ z$
  by *simp*
then have $\forall x.\ \textit{semantics}\ e\ (f(c := \lambda w.\ x))\ g\ (\textit{sub}\ 0\ (\textit{Fun}\ c\ [])\ p)$
  using *Uni-I* **by** *blast*
then have $\forall x.\ \textit{semantics}\ (\textit{put}\ e\ 0\ x)\ (f(c := \lambda w.\ x))\ g\ p$
  by *simp*
then have $\forall x.\ \textit{semantics}\ (\textit{put}\ e\ 0\ x)\ f\ g\ p$
  using $\textit{news}\ c\ (p\ \#\ z)$ **by** *simp*
then show $\textit{semantics}\ e\ f\ g\ (\textit{Uni}\ p)$
  by *simp*

Completeness

Proof by Fitting in *First-Order Logic and Automated Theorem Proving*.
Formalized by Berghofer for different natural deduction proof system.

Dependent on semantics

- Consistency property, C
- Alternative consistency, C^+
- Finite character, C^*
- Maximal extension, H. Is Hintikka, has an Herbrand model

Completeness

Proof by Fitting in *First-Order Logic and Automated Theorem Proving*.
Formalized by Berghofer for different natural deduction proof system.

Dependent on semantics

- Consistency property, C
- Alternative consistency, C^+
- Finite character, C^*
- Maximal extension, H. Is Hintikka, has an Herbrand model

Dependent on inference rules

- Show consistency of formulas from which false cannot be derived.

Completeness

Proof by Fitting in *First-Order Logic and Automated Theorem Proving*.
Formalized by Berghofer for different natural deduction proof system.

Dependent on semantics

- Consistency property, C
- Alternative consistency, C^+
- Finite character, C^*
- Maximal extension, H. Is Hintikka, has an Herbrand model

Dependent on inference rules

- Show consistency of formulas from which false cannot be derived.

Completeness via contradiction

- Assume p is (closed and) valid but not derivable
- Then $\{\neg p\} \in C$ (no contradiction without p), has a model

Standard trick

Standard textbook trick for open formulas: Just universally close it!

$$x \to x \quad \leadsto \quad \forall x.\, x \to x$$

Then we obtain a derivation for a syntactically different formula.

Open formulas are well-defined in our formalization. We should treat them as such.

This might teach students something about environments etc.

Strategy

Starting point $p \overset{?}{\vdash} x \rightarrow p$

Strategy

Starting point	$p \overset{?}{\vdash} x \to p$
Premises to implications	$\overset{?}{\vdash} p \to x \to p$

Strategy

Starting point	$p \overset{?}{\vdash} x \to p$
Premises to implications	$\overset{?}{\vdash} p \to x \to p$
Universally close formula	$\overset{?}{\vdash} \forall x.\, p \to x \to p$

Strategy

Starting point	$p \overset{?}{\vdash} x \to p$
Premises to implications	$\overset{?}{\vdash} p \to x \to p$
Universally close formula	$\overset{?}{\vdash} \forall x.\, p \to x \to p$
Obtain proof	$\vdash \forall x.\, p \to x \to p$

Strategy

Starting point	$p \overset{?}{\vdash} x \to p$
Premises to implications	$\overset{?}{\vdash} p \to x \to p$
Universally close formula	$\overset{?}{\vdash} \forall x.\, p \to x \to p$
Obtain proof	$\vdash \forall x.\, p \to x \to p$
Eliminate quantifiers with constants	$\vdash p \to c \to p$

Strategy

Starting point	$p \overset{?}{\vdash} x \to p$
Premises to implications	$\overset{?}{\vdash} p \to x \to p$
Universally close formula	$\overset{?}{\vdash} \forall x.\, p \to x \to p$
Obtain proof	$\vdash \forall x.\, p \to x \to p$
Eliminate quantifiers with constants	$\vdash p \to c \to p$
Substitute constants with variables	$\vdash p \to x \to p$

Strategy

Starting point	$p \overset{?}{\vdash} x \to p$
Premises to implications	$\overset{?}{\vdash} p \to x \to p$
Universally close formula	$\overset{?}{\vdash} \forall x.\, p \to x \to p$
Obtain proof	$\vdash \forall x.\, p \to x \to p$
Eliminate quantifiers with constants	$\vdash p \to c \to p$
Substitute constants with variables	$\vdash p \to x \to p$
Implications back to premises	$p \vdash x \to p$

Premises to implications

Turn environment into chain of implications:

$\textbf{primrec}\ \textit{put-imps} :: \textit{fm} \Rightarrow \textit{fm list} \Rightarrow \textit{fm}\ \textbf{where}$
$\quad \textit{put-imps}\ p\ [] = p\ |$
$\quad \textit{put-imps}\ p\ (q \mathbin{\#} z) = \textit{Imp}\ q\ (\textit{put-imps}\ p\ z)$

Premises to implications

Turn environment into chain of implications:

primrec *put-imps* :: *fm* ⇒ *fm list* ⇒ *fm* **where**
  put-imps p [] = *p* |
  put-imps p (*q* # *z*) = *Imp q* (*put-imps p z*)

This behaves as expected with regards to the semantics:

lemma *semantics-put-imps*:
  (*list-all* (*semantics e f g*) *z* ⟶ *semantics e f g p*) =
   semantics e f g (*put-imps p z*)
  by (*induct z*) *auto*

Universal closure

Put a number of universal quantifiers in front:

primrec *put-unis* :: *nat* $\Rightarrow$ *fm* $\Rightarrow$ *fm* **where**
  put-unis 0 p $=$ *p* $|$
  put-unis $($*Suc m*$)$ *p* $=$ *Uni* $($*put-unis m p*$)$

Universal closure

Put a number of universal quantifiers in front:

primrec *put-unis* :: *nat* ⇒ *fm* ⇒ *fm* **where**
  put-unis 0 p = *p* |
  put-unis (*Suc m*) *p* = *Uni* (*put-unis m p*)

This preserves validity:

lemma *valid-put-unis*: ∀ (*e* :: *nat* ⇒ ′*a*) *f g*. *semantics e f g p* ⟹
    semantics (*e* :: *nat* ⇒ ′*a*) *f g* (*put-unis m p*)
  by (*induct m arbitrary*: *e*) *simp-all*

Universal closure

Put a number of universal quantifiers in front:

primrec *put-unis* :: *nat* ⇒ *fm* ⇒ *fm* **where**
  put-unis 0 p = *p* |
  put-unis (*Suc m*) *p* = *Uni* (*put-unis m p*)

This preserves validity:

lemma *valid-put-unis*: ∀ (*e* :: *nat* ⇒ *'a*) *f g*. *semantics e f g p* ⟹
    semantics (*e* :: *nat* ⇒ *'a*) *f g* (*put-unis m p*)
  by (*induct m arbitrary*: *e*) *simp-all*

The universal closure exists:

lemma *ex-closure*: ∃ *m*. *sentence* (*put-unis m p*)
  using *ex-closed closed-put-unis* **by** *simp*

Obtain proof

We can combine the above to obtain our derivation:

let *?p* = *put-imps p* (*rev z*)

have ∗: ∀ (*e* :: *nat* ⇒ ′*a*) *f g*. *semantics e f g ?p*
  using *assms semantics-put-imps* **by** *fastforce*
obtain *m* **where** ∗∗: *sentence* (*put-unis m ?p*)
  using *ex-closure* **by** *blast*
moreover have ∀ (*e* :: *nat* ⇒ ′*a*) *f g*. *semantics e f g* (*put-unis m ?p*)
  using ∗ *valid-put-unis* **by** *blast*
ultimately have *OK* (*put-unis m ?p*) []
  using *assms sentence-completeness* **by** *blast*

Next step: Work within proof system to derive open formula from this.

Direct closure elimination

Tricky to eliminate quantifiers directly with de Bruijn indices.

Example

$$
\begin{aligned}
(\forall\forall p(0,1,2))[2/0] \rightsquigarrow \forall((\forall p(0,1,2))[3/1]) \rightsquigarrow \forall\forall(p(0,1,2)[4/2]) &\rightsquigarrow \forall\forall p(0,1,4) \\
(\forall p(0,1,4))[1/0] \rightsquigarrow \forall(p(0,1,4)[2/1]) &\rightsquigarrow \forall p(0,2,3) \\
p(0,2,3)[0/0] &\rightsquigarrow p(0,1,2)
\end{aligned}
$$

Previously substituted variables are adjusted by subsequent substitutions.

Direct closure elimination

Tricky to eliminate quantifiers directly with de Bruijn indices.

Example

$$(\forall\forall p(0,1,2))[2/0] \leadsto \forall((\forall p(0,1,2))[3/1]) \leadsto \forall\forall(p(0,1,2)[4/2]) \leadsto \forall\forall p(0,1,4)$$
$$(\forall p(0,1,4))[1/0] \leadsto \forall(p(0,1,4)[2/1]) \leadsto \forall p(0,2,3)$$
$$p(0,2,3)[0/0] \leadsto p(0,1,2)$$

Previously substituted variables are adjusted by subsequent substitutions.

Idea: Eliminate with (fresh) constants instead!

fun *consts-for-unis* :: *fm* ⇒ *id list* ⇒ *fm* **where**
  consts-for-unis (*Uni p*) (*c*#*cs*) = *consts-for-unis* (*sub 0* (*Fun c* []) *p*) *cs* |
  consts-for-unis p - = *p*

Constant substitution

New type of substitution: *subc c s p* replaces occurences of c with s in p, adjusting s when passing a quantifier.
Disadvantage: Have to reprove many substitution lemmas for *subc*.

Constant substitution

New type of substitution: *subc c s p* replaces occurences of *c* with *s* in *p*, adjusting *s* when passing a quantifier.
Disadvantage: Have to reprove many substitution lemmas for *subc*.
We prove the new rule admissible:

lemma *OK-subc*: $OK\ p\ z \Longrightarrow OK\ (subc\ c\ s\ p)\ (subcs\ c\ s\ z)$

Trivial for everything except cases with quantifiers, newness, e.g. witness in *Exi-E* rule.
Requires renaming:

lemma *OK-psubst*: $OK\ p\ z \Longrightarrow OK\ (psubst\ f\ p)\ (map\ (psubst\ f)\ z)$

Telescoping closure elimination

Composing closure elimination with constant substitution yields telescoping sequence:

$$\textit{subc}\ c_0\ \textit{(m-1)}\ (\textit{subc}\ c_1\ \textit{(m-2)}\ (\ldots\ (\textit{subc}\ c_{m-1}\ \textit{0}\ (\textit{sub}\ \textit{0}\ c_{m-1}\ \ldots))))$$

Each introduced constant is immediately substituted with correct variable. Subsequent substitutions do not adjust previous variables.

lemma *vars-for-consts-for-unis*:
  closed (*length cs*) *p* $\Longrightarrow$ *list-all* (λ*c. new c p*) *cs* $\Longrightarrow$ *distinct cs* $\Longrightarrow$
   vars-for-consts (*consts-for-unis* (*put-unis* (*length cs*) *p*) *cs*) *cs* = *p*

Telescoping closure elimination

Composing closure elimination with constant substitution yields telescoping sequence:

$$\textit{subc}\ c_0\ \textit{(m-1)}\ \textit{(subc}\ c_1\ \textit{(m-2)}\ \textit{(}\dots\ \textit{(subc}\ c_{m-1}\ \textit{0}\ \textit{(sub 0}\ c_{m-1}\ \dots\textit{))))}$$

Each introduced constant is immediately substituted with correct variable. Subsequent substitutions do not adjust previous variables.

lemma *vars-for-consts-for-unis*:
$\textit{closed}\ (\textit{length cs})\ p \Longrightarrow \textit{list-all}\ (\lambda c.\ \textit{new}\ c\ p)\ \textit{cs} \Longrightarrow \textit{distinct cs} \Longrightarrow$
$\textit{vars-for-consts}\ (\textit{consts-for-unis}\ (\textit{put-unis}\ (\textit{length cs})\ p)\ \textit{cs})\ \textit{cs} = p$

theorem *remove-unis*: $\textit{OK}\ (\textit{put-unis}\ m\ p)\ [] \Longrightarrow \textit{OK}\ p\ []$

Implications to premises

For $p \to q$, weaken assumptions with p, then use modus ponens.

lemma *shift-imp-assum*:
  assumes *OK* (*Imp p q*) *z*
  shows *OK q* (*p* # *z*)
proof –
  have *set z* ⊆ *set* (*p* # *z*)
    by *auto*
  then have *OK* (*Imp p q*) (*p* # *z*)
    using *assms weaken-assumptions* **by** *blast*
  moreover have *OK p* (*p* # *z*)
    using *Assume* **by** *simp*
  ultimately show *OK q* (*p* # *z*)
    using *Imp-E* **by** *blast*
qed

Weaken assumptions

lemma *weaken-assumptions*: $OK\ p\ z \Longrightarrow set\ z \subseteq set\ z' \Longrightarrow OK\ p\ z'$

Shown by induction over inference rules.
Trivial, except for *Exi-E* and *Uni-I*, where newness is required: The new constant given by the induction hypothesis is not necessarily new under the bigger premises.
Again, renaming is necessary.

Weaken assumptions

lemma *weaken-assumptions*: $OK\ p\ z \Longrightarrow set\ z \subseteq set\ z' \Longrightarrow OK\ p\ z'$

Shown by induction over inference rules.
Trivial, except for *Exi-E* and *Uni-I*, where newness is required: The new constant given by the induction hypothesis is not necessarily new under the bigger premises.
Again, renaming is necessary.

Remove chain of implications by induction:

lemma *remove-imps*: $OK\ (put\text{-}imps\ p\ z)\ z' \Longrightarrow OK\ p\ (rev\ z\ @\ z')$
using *shift-imp-assum* **by** $(induct\ z\ arbitrary\text{: } z')$ *simp-all*

Completeness

We can now finish the completeness proof:

let $?p = put\text{-}imps\ p\ (rev\ z)$

have $*$: $\forall\,(e :: nat \Rightarrow{} 'a)\ f\ g.\ semantics\ e\ f\ g\ ?p$
  using *assms semantics-put-imps* **by** *fastforce*
obtain m **where** $**$: $sentence\ (put\text{-}unis\ m\ ?p)$
  using *ex-closure* **by** *blast*
moreover have $\forall\,(e :: nat \Rightarrow{} 'a)\ f\ g.\ semantics\ e\ f\ g\ (put\text{-}unis\ m\ ?p)$
  using $*$ *valid-put-unis* **by** *blast*
ultimately have $OK\ (put\text{-}unis\ m\ ?p)\ []$
  using *assms sentence-completeness* **by** *blast*

then have $OK\ ?p\ []$
  using $**$ *remove-unis* **by** *blast*
then show $OK\ p\ z$
  using *remove-imps* **by** *fastforce*

Conclusion

- NaDeA is sound and complete.
- Also for open formulas.
  - Standard results like renaming, weakening arise naturally in proof.
- Formalization ensures tricky cases are treated properly.
- Formalization may also introduce complexity, e.g. de Bruijn indices.

References

TOBIAS NIPKOW, LAWRENCE C. PAULSON AND MARKUS WENZEL, ***Isabelle/HOL — A Proof Assistant for Higher-Order Logic***, vol. 2283, Lecture Notes in Computer Science, Springer, 2002.

STEFAN BERGHOFER, *First-Order Logic According to Fitting*, ***Archive of Formal Proofs***, August 2007. http://isa-afp.org/entries/FOL-Fitting.html

MELVIN FITTING, ***First-Order Logic and Automated Theorem Proving, Second Edition***, Graduate Texts in Computer Science, Springer, 1996.

MARKUS WENZEL, *Isar — A Generic Interpretative Approach to Readable Formal Proof Documents*, ***Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs'99, September, Proceedings*** (Nice, France), (Yves Bertot, Gilles Dowek, André Hirschowitz, Christine Paulin-Mohring and Laurent Théry, editors), vol. 1690, Lecture Notes in Computer Science, Springer, 1999, pp. 167–184.

JØRGEN VILLADSEN, ANDREAS HALKJÆR FROM AND ANDERS SCHLICHTKRULL, *Natural Deduction and the Isabelle Proof Assistant*, Proceedings 6th International Workshop on ***Theorem proving components for Educational software*** (Gothenburg, Sweden), (Pedro Quaresma and Walther Neuper, editors), vol. 267, Electronic Proceedings in Theoretical Computer Science, Open Publishing Association, 2018, pp. 140–155. http://eptcs.org/paper.cgi?ThEdu17.9