Praωf User Manual

Ulrich Berger, Olga Petrovska

2016

Software Requirements *Praωf* has been developed in Haskell and runs within GHCi, the interactive environment of GHC (Glasgow Haskell Compiler). In Windows it is possible to run the software using WinGHCi or run GHCi in shell mode in Emacs. All prover files need to be in the directory `prover`. Additionally, LaTeX is required to display proofs.

Getting started To run the tool in shell mode, follow the below steps:

Move into the directory prover: `cd prover`
Open the Emacs editor: `emacs&` (or `xemacs&`)
Open a shell in Emacs: `M-x shell` (M is a Meta-key or the ESC-key)
Start interactive Haskell: `ghci`
Load the file Prover.hs: `:l Prover.hs`

When using WinGHCi, in order to load the prover click *File* → *Load* and go to the directory where the file Prover.hs is saved and load it.

Run the function main: `main`

In Linux this opens a DVI file displaying the current state of your proof and giving information about the current goal and the available commands.

In Windows you may need to open this file manually. The file name is `pproof.dvi` and it is normally located in the same directory as the prover. All instructions and hints are given at the command prompt and not in the DVI file.

Once a proof is completed the proof tree is written in the file pproof.tex, a LaTeX document.

Connectives & Quantifiers	Possible Input Options			Examples	
$\wedge$	`and`	`&`	`\land`	$A \wedge B$	`A and B`
$\vee$	`or`	`|`	`\lor`	$A \vee B$	`A or B`
$\perp$	`bot` or `Bot` or `F`	`_|_`	`\bot`	$\perp$	`bot`
$\neg$	`not`	`-`	`\neg`	$\neg A$	`not A`
$\forall$	`all` or `All`	`For all`	`\forall`	$\forall x\ A(x)$	`all x A(x)`
$\exists$	`ex` or `Ex`	`Exists`	`\exists`	$\exists x\ A(x)$	`ex x A(x)`
$\to$		`->`	`\to`	$A \to B$	`A -> B`

Syntax of formulas The usual bracketing rules apply when typing in formulas. For example, `A -> (B -> C)` can be written `A -> B -> C`. *Praωf* strips unnecessary parentheses. For example, the input `((A) and (B)) -> ((B) and (A))` will be displayed as $(A \wedge B) \to (B \wedge A)$.

Atomic formula can be any letter except `F`. It also may contain terms. The input format can be `A` or `A(x)`, where `x` is a term or a list of terms. Terms can be either constants, variables or functions (`f(x)`). Terms are separated by commas in the list: `A(x, f(x))`. There should be no space between the predicate and terms.

Composite formulas are built using logical connectives and can be written in various ways as shown in the table below. All input options are case sensitive. Negation can also be written as implication: `not A -> B` is the same as `(not A) -> B` and the same as `(A -> bot) -> B`.
Binding priorities (from strong to weak): `not, and, or, ->`.
Implication, conjunction and disjunction associate to the right.

Example:
`not A -> not B -> A or B and C -> bot`
is the same as
`(not A) -> ((not B) -> ((A or (B and C)) -> bot))`
and also the same as
`(A -> bot) -> (B -> bot) -> not (A or B and C)`

Supported Commands There are two types of commands in *Praωf*: general control commands and specific commands to apply the natural deduction rules.

Control commands	
`undo`	undo a proof step
`quit`	leave the prover
`new`	start a new proof (without saving your current proof)
`submit` i	submit curent proof as solution to question i
`delete` i	delete question i
`?`	more explanations on the commands above

Proof commands		
`use u`	Use an available assumption with label u	
`use`	Same as above. Since the label is missing, you are prompted to enter it.	
`andi`	And introduction rule applied backwards	$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge^{+}$
`andel B`	And elimination left backwards. If the goal was A, the new goal will be $A \wedge B$.	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \wedge_{\mathrm{l}}^{-}$
`andel`	As above, but since the formula B is missing you are prompted to enter it.	
`ander A`	And elimination left backwards. If the goal was B, the new goal will be $A \wedge B$.	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \wedge_{\mathrm{r}}^{-}$
`ander`	As above, but since the formula A is missing you are prompted to enter it.	
`impi u`	Implication introduction backwards. The current goal must be of the form $A \to B$. The new goal is B and A is added as an assumption with a label u.	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \to^{+}$
`impi`	As above, but since the assumption label is missing you are prompted to enter it.	
`impe A`	Implication elimination backwards. If the goal was B, there will be two new goals: $A \to B$ and A.	$\dfrac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B} \to^{-}$
`impe`	As above, but since the formula A is missing you are prompted to enter it.	
`oril`	Or introduction left backwards. The current goal must be of the form $A \vee B$. The new goal is A.	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee_{\mathrm{l}}^{+}$
`orir`	Or introduction right backwards. The current goal must be of the form $A \vee B$. The new goal is B.	$\dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee_{\mathrm{r}}^{+}$

`ore A or B`	Or elimination backwards. If the goal was C, there will be three new goals: $A \vee B$, $A \to C$, and $B \to C$.	$\dfrac{A \vee B \qquad A \to C \qquad B \to C}{C} \vee^-$
`ore`	As above, but since the formula $A \vee B$ is missing you are prompted to enter it.	
`efq`	Ex-falso-quodlibet backwards. The goal can be any formula. The new goal will be $\bot$.	$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}\,\text{efq}$
`raa`	Reductio-ad-absurdum backwards. The goal can be any formula A. The new goal will be the double negation of A.	$\dfrac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}\,\text{raa}$
`alli`	All introduction rule backwards. The current goal must be of the form $\forall x\ A(x)$. The new goal will be $A(x)$. NOTE: x must not occur free in any assumption valid at the point.	$\dfrac{\Gamma \vdash A(x)}{\Gamma \vdash \forall x\, A(x)} \forall^+$
`alle t`	All elimination rule backwards. The current goal must be a predicate formula of the form $A(t)$[1], where t is a term.	$\dfrac{\Gamma \vdash \forall x\, A(x)}{\Gamma \vdash A(t)} \forall^-$
`alle x`	All elimination rule backwards. The current goal must be a predicate formula of the form $A(x)$, where x is a term.	$\dfrac{\Gamma \vdash \forall x\, A(x)}{\Gamma \vdash A(x)} \forall^-$
`alle`	As above, but since the term is missing you are prompted to enter it.	
`exi t`	Exists introduction rule backwards. The current goal must be of the form $\exists x\ A(x)$.[1] You need to add a term t which will subsitute the variable x.	$\dfrac{\Gamma \vdash A(t)}{\Gamma \vdash \exists x\, A(x)} \exists^+$
`exi`	As above, but since the term is missing you are prompted to enter it.	
`exe ex x A(x)`	Exists elimination rule backwards. The current goal can be any formula. NOTE: x must not be free in B.	$\dfrac{\exists x\, A(x) \qquad \forall x\,(A(x) \to B)}{B} \exists^-$

`exe`	As above, but since the quantified formula is missing you are prompted to enter it.	

Example Session in GHCi

```
GHCi, version 8.0.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Lenovo\Desktop\Haskell\Prover_files\prover
Prelude> :load "Prover.hs"
[ 1 of 10] Compiling MapAux           ( MapAux.hs, interpreted )
[ 2 of 10] Compiling SystemW          ( SystemW.hs, interpreted )
[ 3 of 10] Compiling Perhaps          ( Perhaps.hs, interpreted )
[ 4 of 10] Compiling Parser           ( Parser.hs, interpreted )
[ 5 of 10] Compiling Formula          ( Formula.hs, interpreted )
[ 6 of 10] Compiling Proof            ( Proof.hs, interpreted )
[ 7 of 10] Compiling Buss             ( Buss.hs, interpreted )
[ 8 of 10] Compiling Step             ( Step.hs, interpreted )
[ 9 of 10] Compiling ReadShow         ( ReadShow.hs, interpreted )
[10 of 10] Compiling Prover           ( Prover.hs, interpreted )
Ok, modules loaded: Parser, Prover, Perhaps, Formula, Proof, Buss, Step,
ReadShow, SystemW, MapAux.
*Prover> main
Enter goal formula X > ex x (A(x)) -> all x (A(x) -> B(x)) -> ex x B(x)
Enter command> impi u1
Enter command> impi u2
Enter command> exe ex x A(x)
Enter command> use u1
Enter command> alli
Enter command> impi u3
Enter command> exi x
Enter command> impe A(x)
Enter command> alle all x (A(x) -> B(x))
Enter command> use u2
Enter command> use u3
Proof complete.
Enter quit, submit <i>, delete <i>, new, or ?> quit
*Prover>
```

This session generates the following proof tree:

[1]By $A(t)$ we mean any formula containing the term t, not just the application of the predicate A to the term t.

$$
\begin{prooftree}
\AxiomC{$u1 : \exists x\ A(x)$}
\AxiomC{$u2 : \forall x\ (A(x) \to B(x))$}
\RightLabel{$\forall^-$}
\UnaryInfC{$A(x) \to B(x)$}
\AxiomC{$u3 : A(x)$}
\RightLabel{$\to^-$}
\BinaryInfC{$B(x)$}
\RightLabel{$\exists^+$}
\UnaryInfC{$\exists x\ B(x)$}
\RightLabel{$\to^+\ u3 : A(x)$}
\UnaryInfC{$A(x) \to \exists x\ B(x)$}
\RightLabel{$\forall^+$}
\UnaryInfC{$\forall x\ (A(x) \to \exists x\ B(x))$}
\RightLabel{$\exists^-$}
\BinaryInfC{$\exists x\ B(x)$}
\RightLabel{$\to^+\ u2 : \forall x\ (A(x) \to B(x))$}
\UnaryInfC{$\forall x\ (A(x) \to B(x)) \to \exists x\ B(x)$}
\RightLabel{$\to^+\ u1 : \exists x\ A(x)$}
\UnaryInfC{$\exists x\ A(x) \to (\forall x\ (A(x) \to B(x)) \to \exists x\ B(x))$}
\end{prooftree}
$$

Some more examples

Proof of $\forall x\ (A(x) \to B(x)) \to (\forall x\ A(x) \to \forall x\ B(x))$

```
Enter goal formula X > all x (A(x) -> B(x)) -> all x (A(x)) -> All x (B(x))
Enter command> impi u1
Enter command> impi u2
Enter command> alli
Enter command> impe
Enter missing formula X> A(x)
Enter command> alle
Enter the term you wish to generalise> x
Enter command> use u1
Enter command> alle
Enter the term you wish to generalise> x
Enter command> use u2
Proof complete.
```

$$
\begin{prooftree}
\AxiomC{$u1 : \forall x\ (A(x) \to B(x))$}
\RightLabel{$\forall^-$}
\UnaryInfC{$A(x) \to B(x)$}
\AxiomC{$u2 : \forall x\ A(x)$}
\RightLabel{$\forall^-$}
\UnaryInfC{$A(x)$}
\RightLabel{$\to^-$}
\BinaryInfC{$B(x)$}
\RightLabel{$\forall^+$}
\UnaryInfC{$\forall x\ B(x)$}
\RightLabel{$\to^+\ u2 : \forall x\ A(x)$}
\UnaryInfC{$\forall x\ A(x) \to \forall x\ B(x)$}
\RightLabel{$\to^+\ u1 : \forall x\ (A(x) \to B(x))$}
\UnaryInfC{$\forall x\ (A(x) \to B(x)) \to (\forall x\ A(x) \to \forall x\ B(x))$}
\end{prooftree}
$$

Proof of $\forall x\ (A(x) \wedge B(x)) \to \forall x\ A(x)$

```
Enter goal formula X > all x (A(x) and B(x)) -> all x A(x)
Enter command> impi u1
Enter command> alli
Enter command> andel
Enter missing formula X> B(x)
Enter command> alle
Enter the term you wish to generalise> x
Enter command> use u1
Proof complete.
```

$$
\begin{prooftree}
\AxiomC{$u1 : \forall x\ (A(x) \wedge B(x))$}
\RightLabel{$\forall^-$}
\UnaryInfC{$A(x) \wedge B(x)$}
\RightLabel{$\wedge^-\mathrm{l}$}
\UnaryInfC{$A(x)$}
\RightLabel{$\forall^+$}
\UnaryInfC{$\forall x\ A(x)$}
\RightLabel{$\to^+\ u1 : \forall x\ (A(x) \wedge B(x))$}
\UnaryInfC{$\forall x\ (A(x) \wedge B(x)) \to \forall x\ A(x)$}
\end{prooftree}
$$

Proof of $(\forall x\ A(x) \wedge \forall x\ B(x)) \to \forall x\ (A(x) \wedge B(x))$

```
Enter goal formula X > all x A(x) and all x B(x) -> all x (A(x) and B(x))
Enter command> impi u1
Enter command> alli
Enter command> andi
Enter command> alle
Enter the term you wish to generalise> x
Enter command> andel
Enter missing formula X> all x B(x)
Enter command> use u1
Enter command> alle
Enter the term you wish to generalise> x
Enter command> ander
Enter missing formula X> all x A(x)
Enter command> use u1
Proof complete.
```

$$
\begin{prooftree}
\AxiomC{$u1 : \forall x\ A(x) \wedge \forall x\ B(x)$}
\RightLabel{$\wedge^-\mathrm{l}$}
\UnaryInfC{$\forall x\ A(x)$}
\RightLabel{$\forall^-$}
\UnaryInfC{$A(x)$}
\AxiomC{$u1 : \forall x\ A(x) \wedge \forall x\ B(x)$}
\RightLabel{$\wedge^-\mathrm{r}$}
\UnaryInfC{$\forall x\ B(x)$}
\RightLabel{$\forall^-$}
\UnaryInfC{$B(x)$}
\RightLabel{$\wedge^+$}
\BinaryInfC{$A(x) \wedge B(x)$}
\RightLabel{$\forall^+$}
\UnaryInfC{$\forall x\ (A(x) \wedge B(x))$}
\RightLabel{$\to^+\ u1 : \forall x\ A(x) \wedge \forall x\ B(x)$}
\UnaryInfC{$(\forall x\ A(x) \wedge \forall x\ B(x)) \to \forall x\ (A(x) \wedge B(x))$}
\end{prooftree}
$$

Proof of $(\exists x\ (A(x) \to \bot) \to \bot) \to \forall x\ A(x)$

```
Enter goal formula X > not ex x (not A(x)) -> all x A(x)
Enter command> impi u1
Enter command> alli
Enter command> raa
Enter command> impi u2
Enter command> exe
Enter missing formula of the form: ex x A(x)> ex x (A(x) -> F)
Enter command> exi
Enter a term that should substitute the variable> x
Enter command> use u2
Enter command> efq
Enter command> impe
Enter missing formula X> ex x (A(x) -> F)
Enter command> use u1
Enter command> exi
Enter a term that should substitute the variable> x
Enter command> use u2
Proof complete.
```

$$
\begin{prooftree}
\AxiomC{$u2 : A(x) \to \bot$}
\RightLabel{$\exists^+$}
\UnaryInfC{$\exists x\ (A(x) \to \bot)$}
\AxiomC{$u1 : \exists x\ (A(x) \to \bot) \to \bot$}
\AxiomC{$u2 : A(x) \to \bot$}
\RightLabel{$\exists^+$}
\UnaryInfC{$\exists x\ (A(x) \to \bot)$}
\RightLabel{$\to^-$}
\BinaryInfC{$\bot$}
\RightLabel{efq}
\UnaryInfC{$\forall x\ ((A(x) \to \bot) \to \bot)$}
\RightLabel{$\exists^-$}
\BinaryInfC{$\bot$}
\RightLabel{$\to^+\ u2 : A(x) \to \bot$}
\UnaryInfC{$(A(x) \to \bot) \to \bot$}
\RightLabel{raa}
\UnaryInfC{$A(x)$}
\RightLabel{$\forall^+$}
\UnaryInfC{$\forall x\ A(x)$}
\RightLabel{$\to^+\ u1 : \exists x\ (A(x) \to \bot) \to \bot$}
\UnaryInfC{$(\exists x\ (A(x) \to \bot) \to \bot) \to \forall x\ A(x)$}
\end{prooftree}
$$

Proof of $\exists x\ (A(x) \wedge B(x)) \to (\exists x\ A(x) \wedge \exists x\ B(x))$

```
Enter goal formula X > ex x (A(x) and B(x)) -> ex x A(x) and ex x B(x)
Enter command> impi u1
Enter command> exe
Enter missing formula of the form: ex x A(x)> ex x (A(x) and B(x))
Enter command> use u1
Enter command> alli
Enter command> impi u2
Enter command> andi
Enter command> exi
```

```
Enter a term that should substitute the variable> x
Enter command> andel
Enter missing formula X> B(x)
Enter command> use u2
Enter command> exi
Enter a term that should substitute the variable> x
Enter command> ander
Enter missing formula X> A(x)
Enter command> use u2
Proof complete.
```

$$
\dfrac{u1 : \exists x\ (A(x) \wedge B(x)) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{u2 : A(x) \wedge B(x)}{A(x)}\,\wedge^{-}\mathrm{l}}{\exists x\ A(x)}\,\exists^{+} \qquad \dfrac{\dfrac{u2 : A(x) \wedge B(x)}{B(x)}\,\wedge^{-}\mathrm{r}}{\exists x\ B(x)}\,\exists^{+}}{\exists x\ A(x) \wedge \exists x\ B(x)}\,\wedge^{+}}{(A(x) \wedge B(x)) \to (\exists x\ A(x) \wedge \exists x\ B(x))}\,\to^{+}\ u2 : A(x) \wedge B(x)}{\forall x\ ((A(x) \wedge B(x)) \to (\exists x\ A(x) \wedge \exists x\ B(x)))}\,\forall^{+}}{\dfrac{\exists x\ A(x) \wedge \exists x\ B(x)}{\exists x\ (A(x) \wedge B(x)) \to (\exists x\ A(x) \wedge \exists x\ B(x))}\,\to^{+}\ u1 : \exists x\ (A(x) \wedge B(x))}\,\exists^{-}
$$

`THESE ARE PROOFS WITH BUGS/must be something wrong in parsing`

When proving $\exists x\ (A(x) \to \bot) \to (\forall x\ A(x) \to \bot)$ the 3rd step defines if it is provable or not. If you use $A(\mathrm{t})$ for term and not provable if you use x and not t, then at the last step when alle is applied and you decide to quantify x, then it gets the following form $All\ x(A(\mathrm{x})$ and becomes unprovable.

Provable version:

$$
\begin{prooftree}
\AxiomC{$u1 : \exists x\ (A(x) \to \bot)$}
\AxiomC{$u3 : A(x) \to \bot$}
\AxiomC{$u2 : \forall x\ A(x)$}
\RightLabel{$\forall^-$}
\UnaryInfC{$A(x)$}
\RightLabel{$\to^-$}
\BinaryInfC{$\bot$}
\RightLabel{$\to^+\ u4 : A(\mathrm{t})$}
\UnaryInfC{$A(\mathrm{t}) \to \bot$}
\RightLabel{$\to^+\ u3 : A(x) \to \bot$}
\UnaryInfC{$(A(x) \to \bot) \to (A(\mathrm{t}) \to \bot)$}
\RightLabel{$\forall^+$}
\UnaryInfC{$\forall x\ ((A(x) \to \bot) \to (A(\mathrm{t}) \to \bot))$}
\RightLabel{$\exists^-$}
\BinaryInfC{$A(\mathrm{t}) \to \bot$}
\AxiomC{$u2 : \forall x\ A(x)$}
\RightLabel{$\forall^-$}
\UnaryInfC{$A(\mathrm{t})$}
\RightLabel{$\to^-$}
\BinaryInfC{$\bot$}
\RightLabel{$\to^+\ u2 : \forall x\ A(x)$}
\UnaryInfC{$\forall x\ A(x) \to \bot$}
\RightLabel{$\to^+\ u1 : \exists x\ (A(x) \to \bot)$}
\UnaryInfC{$\exists x\ (A(x) \to \bot) \to (\forall x\ A(x) \to \bot)$}
\end{prooftree}
$$

Unprovable version:

$$
\begin{prooftree}
\AxiomC{$u1 : \exists x\ (A(x) \to \bot)$}
\AxiomC{$u3 : A(x) \to \bot$}
\AxiomC{$u2 : \forall x\ A(x)$}
\RightLabel{$\forall^-$}
\UnaryInfC{$A(x)$}
\RightLabel{$\to^-$}
\BinaryInfC{$\bot$}
\RightLabel{$\to^+\ u4 : A(\mathrm{x})$}
\UnaryInfC{$A(\mathrm{x}) \to \bot$}
\RightLabel{$\to^+\ u3 : A(x) \to \bot$}
\UnaryInfC{$(A(x) \to \bot) \to (A(\mathrm{x}) \to \bot)$}
\RightLabel{$\forall^+$}
\UnaryInfC{$\forall x\ ((A(x) \to \bot) \to (A(\mathrm{x}) \to \bot))$}
\RightLabel{$\exists^-$}
\BinaryInfC{$A(\mathrm{x}) \to \bot$}
\AxiomC{$?\mathbf{13} : \forall x\ A(\mathrm{x})$}
\RightLabel{$\forall^-$}
\UnaryInfC{$A(\mathrm{x})$}
\RightLabel{$\to^-$}
\BinaryInfC{$\bot$}
\RightLabel{$\to^+\ u2 : \forall x\ A(x)$}
\UnaryInfC{$\forall x\ A(x) \to \bot$}
\RightLabel{$\to^+\ u1 : \exists x\ (A(x) \to \bot)$}
\UnaryInfC{$\exists x\ (A(x) \to \bot) \to (\forall x\ A(x) \to \bot)$}
\end{prooftree}
$$

There is a problem with exi because when you apply it to, for example $ExxA(x)$, sometimes you get $A(x)$ but sometimes you get $A(\mathtt{x})$. So in the proof of $\exists x\ (A(x) \wedge B(x)) \to (\exists x\ A(x) \wedge \exists x\ B(x))$ above you get $A(x)$ but when I do for $(\forall x\ (A(x) \to \bot) \to \bot) \to \exists x\ A(x)$ it changes to $A(\mathtt{x})$. I cannot quite understand what causes this different befaviour.

$$
\dfrac{\dfrac{?\mathbf{2} : A(\mathrm{x})}{\exists x\ A(x)}\ \exists^+}{(\forall x\ (A(x) \to \bot) \to \bot) \to \exists x\ A(x)} \to^+\ u1 : \forall x\ (A(x) \to \bot) \to \bot
$$