第57回 プログラミング･シンポジウム 2016.1.8-10

定理証明支援系のGUIにおける証明木表示機能の拡張

Extension of a Proof Assistant's GUI for Displaying Traditional Proof Trees

田中 雄太[1,a)]　川端 英之[1]　北村 俊明[1]

概要：ソフトウェアの検証や数学的な定理証明のために，定理証明支援系が利用されている．しかし，既存の定理証明系及びそのユーザインタフェースには，初学者には使い難い点がある．例えば，定理証明支援系 Coq などのインタフェースである ProofGeneral の備える証明木可視化ツール Prooftree は，極度に簡略化された証明木しか表示する機能を持たない．本研究では，Prooftree に注目し，定理証明支援系の初学者や，数理論理学で一般に用いられる証明木に慣れているユーザを対象に，より分かり易い木を表示する機能や，証明の進行を制御する機能を導入する．これにより，ProofGeneral に慣れていないユーザや，標準証明木に慣れているユーザに対し，より効果的に証明の支援を行えると期待できる．

キーワード：定理証明支援系，証明木，Coq，ProofGeneral，GUI

1. はじめに

ソフトウェアの検証や数学的な定理証明のために，定理証明支援系が利用されている．定理証明支援系には，Coq[1]，Agda[2]，Isabelle/HOL[3] などがあり，例えば，Coq を用いた四色定理の形式化 [4] や，C 言語コンパイラの CompCert の健全性の証明 [5] が行われている．

定理証明支援系を利用し，証明や型検査を Emacs 上で行うためのインタフェースとして ProofGeneral[6] がある (図 1)．ProofGeneral には，ユーザの証明の支援を行う証明木の可視化ツール Prooftree[7] が備えられている．Prooftree は，証明時のステップ毎のサブゴールや，タクティクコマンドを描画する機能がある (図 2)．

証明木は，証明の挙動を見る際に利用される．ユーザは証明木を眺め，次に適用するタクティクを思案し，証明を進める．ユーザは，Prooftree を利用すると，各ステップ毎のシークエントを確認しながら証明を進めることができる．しかし，Prooftree は，ProofGeneral に使い慣れていないユーザや，数理論理学で一般に用いられる証明木（以下，本稿では標準証明木と呼ぶ）に慣れているユーザに対し，効果的に証明の支援を行えているとは言い難い．それは，以下のような理由である．

[1] 広島市立大学 大学院情報科学研究科
[a)] ma66018@e.hiroshima-cu.ac.jp

- 各証明ステップ前後でのサブゴールや仮定の変化を確認し辛い
- 標準証明木との対応が取り辛い
- 描画する機能のみであり，証明を進めることはできない

本研究では，定理証明支援系のユーザがより効果的に証明を進められるよう，Prooftree 及び ProofGeneral に拡張を施す (なお，本研究では定理証明支援系 Coq の利用を想定する)．上記に挙げた問題を解決するために，以下の 2 つの機能を導入する．

(1) 数理論理学で一般に用いられる証明木のように下から上へ枝が伸び，各ステップでの仮定やサブゴールを一目で確認できるような証明木を表示する機能

(2) Prooftree 側から証明を進められる機能

本稿では，拡張を施すシステムとその関係について述べ，既存ツールの課題点と拡張の提案，課題に対する拡張設計，拡張の現状について説明する．

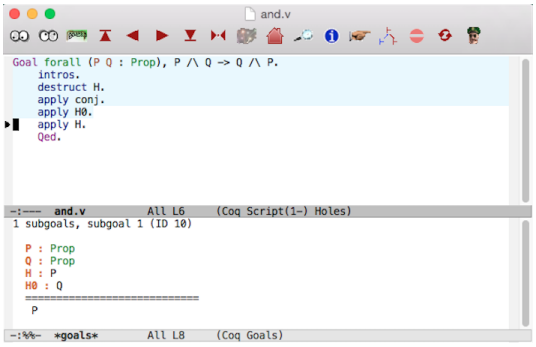

図 1 ProofGeneral の GUI

2. 定理証明支援系 Coq と関連ツール

定理証明支援系は，ユーザの証明の記述を支援し，プログラムの正しさを保証する際に利用される．定理証明支援系は，証明のステップのためのスクリプトの記述を受け取ると，元々のゴールへたどり着くための新たなゴールを返し，対話的に証明を構成する．

定理証明支援系のひとつに，Coq が挙げられる．ユーザは Coq を利用して，Gallina[8] で記述した関数を，命題やタクティクを記述して証明を行う．また，定義された関数を OCaml や Haskell，Scheme などの，他の言語で記述された関数へ変換する機能も備えられている．そのため，所望の性質を持つことが証明されたコードを，他の言語で作られたプログラムに導入することができる．Coq のインタフェースには，Emacs 上で動作する ProofGeneral，CoqIDE，jEdit などがあり，ProofGeneral には証明木可視化のためのツール Prooftree が備えられている．

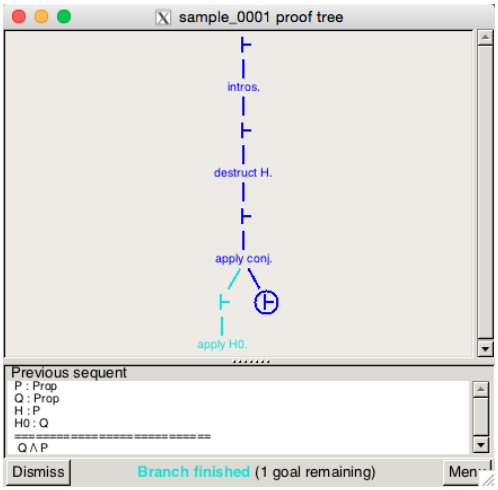

図 2 Prooftree の GUI

2.1 ProofGeneral

ProofGeneral は，図 1 のような GUI でユーザの証明の支援を行う定理証明支援系の開発環境である．ProofGeneral は Emacs Lisp で記述され，Emacs 上で証明を行う際に利用される．

ProofGeneral には，ユーザが記述した命題やタクティクを評価し，各ステップ毎のサブゴールと仮定を表示する機能が備えられている (図 1)．ユーザは同図中の`*goals*`バッファに示されたサブゴールをもとに，次のステップへのタクティクを記述しながら証明を進めていく．

2.2 Prooftree

Prooftree は ProofGeneral 上での証明時に，証明木の可視化に利用される外部インタフェースで

ある (図 2)．Prooftree は，OCaml で記述されており，GUI ライブラリである Lablgtk[9] を利用してウィンドウなどの表示を行う．

ProofGeneral に用意された proof-tree ボタンをクリックすることで，Prooftree のウィンドウが起動する．Prooftree は，ProofGeneral から送られた証明ステップ毎のシークエント情報や，タクティクコマンドをもとに，そのステップにおける証明木を描画する．ユーザが，簡略化された証明木の節をクリックすると，そのステップにおけるサブゴールが，図 2 の下段に配置された `sequent` エリアに表示される．

2.3 システム構成とその特徴

上記に挙げた Coq と関連ツールの構成を図 3 に示す．ProofGeneral は，Emacs 上で対話型証明を行う際に利用される．ProofGeneral は，ユーザが記述した定理やタクティクを受け取り，それらの情報を Coq へ伝える．Coq は，ProofGeneral から受け取ったタクティクが適用できるかを判断し，適用できる際には新たなサブゴールを生成し，ProofGeneral へ伝える．また，適用できない際も同様に，その情報を ProofGeneral へ伝える．

Prooftree は，対話型証明を行う際に，証明木の可視化を行い，ユーザの証明のサポートを行うツールである．Prooftree は，ProofGeneral から送られた情報をもとに，証明木を描画する機能をもつ．しかし，Prooftree から ProofGeneral へ情報を送ることはない．同期の問題を避けるために，双方向の通信は行わない仕様とされている．

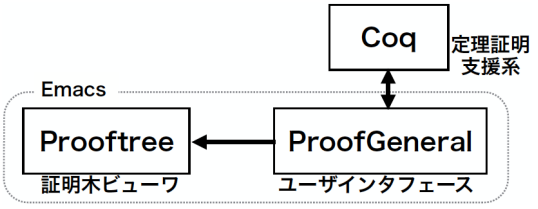

図 3 Coq とその関連ツールの構成

3. 定理証明支援系の GUI の拡張の提案

本章では，前章で説明した既存の Prooftree 及び ProofGeneral の課題点の詳細を述べ，その課題点に対する拡張方針について説明する．

3.1 Prooftree-ProofGeneral の問題点

現在の Prooftree では，ProofGeneral を使い慣れていないユーザや，標準証明木に慣れているユーザに対し，効果的に証明の支援を行えているとは言い難い点がある．それは，以下のような理由である．

(1) 各証明ステップ前後でのサブゴールや仮定の変化を確認し辛い
(2) 標準証明木との対応が取り辛い
(3) 描画する機能のみであり，証明を進めることはできない

これら 3 つの課題の詳細を順に説明する．

3.1.1 ステップ毎の状況変化把握の難しさ

(1) の問題は，サブゴールとタクティクの 2 つの問題に分けられる．

サブゴールの表示の問題は，例えば図 2 を眺めた際に，`destruct H.` タクティクが適用される前のサブゴールと，適用後のサブゴールを一目では確認できない点である．現在の Prooftree では，`destruct H.` タクティクの上下にある⊢アイコンをクリックしなければ，その時点でのサブゴールは表示されない．また，ある 2 つのステップにおけるサブゴールを同時に表示させて，タクティク適用によるサブゴールの変化を一目で確認することはできない．

タクティクの表示の問題は，図 2 のように，表示されている木を眺めた際に，タクティクのノードの意味しか分からない点である．現在の Prooftree は，木のノードとしては，図中⊢で表されたサブゴールのノードと，`intros.` や `destruct H.` と表されたタクティクのノードの 2 種類のみしか表示されていない．そのため，`destruct H.` が適用される際の，`H` という識別子が添えられた仮定の内容を確認するには，`destruct H.` タクティクのノードのひとつ上にある⊢アイコンをクリックし，`sequent` エリアの仮定を見なければならない．また，`destruct H.` タクティク適用前後で，`H` という識別子が添えられた仮定が，どのように変わった

かについても，一目では確認することができない．

3.1.2 表示される証明木の形

この問題は，現在の Prooftree の木の形が，標準証明木における木の形と異なる点である．一般に，証明木と呼称される木は，図 4 のウィンドウ中に表示されたような木の形をとる．最下部にゴールとなるノードがあり，そのノードのサブゴールがひとつ上段へ記述されている．また，サブゴールの変化の際に適用したタクティクを，棒線の隣に記述するのが慣習である．標準証明木は，証明のステップを進めていくと，最下段にあるノードから，順に上へ枝が伸びていく．これに対し，現在の Prooftree では，ゴールとなるノードは最上段にあり，証明のステップを進めると，枝は下に伸びていく．

3.1.3 証明進行の制御

この問題は，現在の Prooftree が，ProofGeneral から受け取った情報から証明木を表示する機能しか持たない点である．Prooftree は，ProofGeneral で進行中の証明の情報を，証明が 1 ステップ進むタイミングで取得する．取得した情報をもとに，証明木を変化させ，新たな証明木を描画する．証明を進めるのは ProofGeneral であり，Prooftree は進行状況をユーザに証明木の描画で伝える役割しかない．ユーザは Prooftree に描画された証明木やシークエント情報等を見ながら，ProofGeneral の UI に証明を記述する必要があるため，ウィンドウの行き来といった手間がかかってしまい，効率的ではない．

3.2 定理証明支援系の GUI の拡張方針

本研究では，定理証明支援系のユーザがより効果的に証明を進められるよう，Prooftree 及び ProofGeneral に拡張を施し，3.1 節で述べた既存ツールの課題点を克服する．

本研究では，(1) 数理論理学で一般に用いられる証明木のように下から上へ枝が伸び，各ステップでのサブゴールを一目で確認できるような証明木を表示する機能を Prooftree に導入することを検討する．この機能は，各ステップにおける変化を確認できるよう，タクティク適用前のサブゴールを表示させ，どのタクティクや仮定を利用し，サブゴールが変化したのかを明示し，標準証明木のように木を変化させて表示することで，上記に挙げた問題 (1)，(2) を解決する．また，上記に挙げた問題 (3) を解決するために，(2)Prooftree 側から証明を進められる機能の導入を検討する．Prooftree のウィンドウ中に証明を進める機能を導入することで，ユーザは ProofGeneral と Prooftree のウィンドウの行き来することなく証明を進められる．

4. 定理証明支援系の GUI の拡張計画とその現状

本章では，3.2 節で述べた拡張方針の詳細と，現在までの取り組みについて説明する．

4.1 標準証明木表示機能の設計

標準証明木表示機能は，数理論理学で一般に用いられる証明木のように下から上へ枝が伸び，各ステップ毎のサブゴールや仮定を一目で確認できるような証明木を表示する機能である．新たに標準証明木表示のための新たなウィンドウを追加し，図 4 のような証明木の形でシークエント，タクティクコマンド，仮定を表示できるよう拡張を施す．これにより，ユーザは最初のゴールからどのようなタクティクコマンドや仮定を適用し，サブゴールが変化したかを一目で確認でき，各ステップにおけるサブゴールも確認できる．

標準証明木表示機能の実現には，ProofGeneral から送られるメッセージを蓄積し，その情報をもとに，標準証明木のためのデータ構造を作り出す必要がある．そのために，Prooftree が受け取ったメッセージの構造を調査し，標準証明木の移り変わりの様子を把握し，適切に表示する機能を導入する．

ProofGeneral からのメッセージは，ステップが進むタイミングと同時に送られる．送られたメッセージは，木の名前や適用したタクティクコマンド，シークエントなどの情報が 1 つのデータにまとめられている．そのため，各情報を分割し，標準証明木のデータ構造に合わせて情報を蓄積する必要がある．また，標準証明木は，適用するタク

ティクによって木の形が変化するため，タクティクによって木構造を変化させ，適切に木を表示する機能を導入する．

現在までに，標準証明木表示機能の導入に向けて，以下の取り組みを行った．

- 標準証明木のデータ構造の設計
- Prooftree が ProofGeneral から受け取る情報の取り出し
- 取り出した情報の証明木用データ構造への成形
- 木構造変化に向けたタクティクと推論規則との対応の調査
- 一部タクティクの適用時のウィンドウへの木の情報の出力

今後は，タクティクごとのデータ構造の変化の設計と構築を行い，蓄積した木構造のデータを，適切に標準証明木の形でウィンドウに表示できるよう実装を行う．

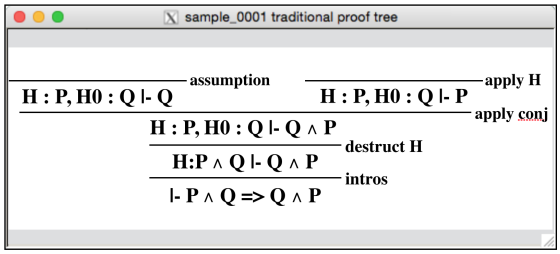

図 4　標準証明木ウィンドウのイメージ

4.2　証明を進める機能の設計

証明を進める機能は，標準証明木ウィンドウからタクティクコマンドの入力を行い，証明のステップを進められる機能である．図 5 のように，標準証明木ウィンドウ中に，次のステップへ進めるために必要となるタクティク箇所をクリックできるようにする．クリックした際にどのタクティクを適用するかを選択するために，新たに**コマンド選択ウィンドウ**を表示させる．また，選択したタクティクが仮定を必要とする場合には，コマンドクリックの際に，**仮定選択ウィンドウ**も表示させるよう拡張を施す．

例えば，図 5 の証明のステップの時点では，`apply conj`．タクティクによるサブゴール `P`，`Q` の 2 箇所について証明する必要がある．ユーザは，サブゴール `P` の証明を進めたい場合には，図中サブゴール `P` の箇所の `tactic` ボタンをクリックする．クリックにより生成されたコマンド選択ウィンドウから，サブゴール `P` に適用させたいタクティクをクリックする．もし選択したタクティクが仮定を必要とする場合には，クリックにより生成された仮定選択ウィンドウから，利用する仮定をクリックする．これにより，ユーザはどのステップでどのタクティクが適用され，サブゴールが変化したかの一式の情報を一目で確認しながら，タクティクコマンドを入力でき，より効果的に証明を進めることができる．

実現にあたり，選択したタクティクコマンドが必要とする仮定は，ステップによって変化するので，ステップ毎に利用する仮定を変化させて表示させる機構を作る必要がある．また，Prooftree だけではなく，ProofGeneral に対しても拡張を施す必要がある．現在の Prooftree では，ProofGeneral からメッセージを受け取り，表示する機能しか持たないので，Prooftree と ProofGeneral 間を双方向でメッセージの授受を行わなければならない．そのために，Prooftree には，ユーザが選択したタクティクを ProofGeneral へ送信する機構を作り，ProofGeneral には，Prooftree から送られたタクティクを受信する機構を作る．また，ProofGeneral には，Prooftree から受け取ったタクティクを，ProofGeneral 上で証明を進める際に利用するタクティクと同様に扱うための機構も作る．

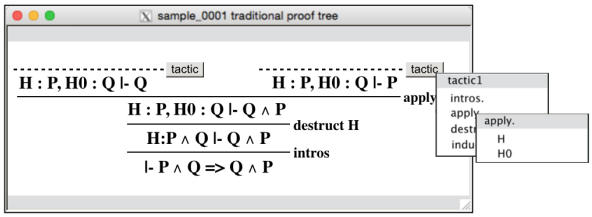

図 5　証明進行機能付きウィンドウのイメージ

5.　関連研究

様々なアプローチから証明の支援を行うための

研究がなされている．ここでは，以下の 2 つの証明の支援に向けた研究を挙げ，これらの研究の概要を説明し，本研究との相違点を述べる．

5.1 Miki β [10]

型推論等で用いられる証明木は，型システムを用いた項の型付けを検証したり，その挙動を見るために描かれる．しかし，手で証明木を描く際には推論規則の本質とは関係のないところで労力を割かなければならない．このシステムでは，証明木を作成する GUI の構築を支援する．Miki β はユーザが記述した式と適用する推論規則を入力として受け取り，推論規則が適用できるか判定し，その証明木を表示する機能をもつ．

Miki β は型推論等で用いられる証明木を利用するユーザを対象としているが，本研究では，定理証明支援系 Coq を用いて証明を行うユーザに向けた証明木の表示を目指している．

5.2 Tree Proof Generator[11]

Tree Proof Generator は，JavaScript で記述された証明木作成ツールである．このツールで描画される証明木は，古典的な命題や述語論理の証明木を対象としている．Tree Proof Generator は，ユーザが記述した命題を入力として受け取り，その命題が成り立つ場合には，証明完了までの木を逐次的に表示する．受け取った命題が成り立たない場合には，invalid を返す．

Tree Proof Generator は，入力として受け取るのは命題のみで，システム自身で証明を構成し，その証明木を表示する．また，証明の各ステップでどのような推論規則が適用されたかなどを，ユーザは把握することができない．本研究では，ユーザ自身が推論を行い，各ステップでの仮定やサブゴールを確認できる証明木を描画し，証明の支援を行う．

6. まとめ

本研究では，定理証明支援系の既存ツールの課題を克服するために，標準証明木表示機能と，証明を進める機能を導入する．これにより，ProofGeneral に慣れていないユーザや，数理論理学で一般に用いられる証明木に慣れているユーザに対し，より効果的に証明の支援を行う．

標準証明木表示機能の導入に向けた今後の取り組みは，タクティクごとのデータ構造の変化の設計と構築を行い，適切に標準証明木の形でウィンドウに表示させることである．

証明を進める機能の導入にむけた今後の取り組みは，Prooftree と ProofGeneral 間のプロトコルの設計と実装を行い，Prooftree から受け取ったタクティクを既存のタクティクと同様に扱うための機構を導入することである．

参考文献

[1] The Coq Proof Assistant, https://coq.inria.fr/.

[2] Agda, http://ocvs.cfv.jp/Agda/.

[3] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel, “Isabelle/HOL — A Proof Assistant for Higher-Order Springer, 2002.

[4] Georges Gonthier, “A computer-checked proof of the Four Colour Theorem,” http://research.microsoft.com/en-US/people/gonthier/4colproof.pdf.

[5] CompCert, http://compcert.inria.fr/.

[6] Proof General, http://proofgeneral.inf.ed.ac.uk/.

[7] Prooftree, http://askra.de/software/prooftree/.

[8] The Gallina specification language, https://coq.inria.fr/refman/Reference-Manual003.html.

[9] LablGtk, http://lablgtk.forge.ocamlcore.org/.

[10] MikiBeta, http://www.is.ocha.ac.jp/ asai/MikiBeta/.

[11] Tree Proof Generator, http://www.umsu.de/logik/trees/info.html.