Point-and-Click Logic

Matti Nykänen Raul Hakli

Department of Computer Science
University of Helsinki
Finland

2nd June 2004

Summary of the Talk

We present. . .

- a user interface concept
- for proof editors
- where the mouse is enough for generating the logical proof
- which can be used in certain kinds of sequent calculi.

We demonstrate an implementation of this concept.

The Trinity of Logics

Model theory:

- Semantics of the logical language.
- "Is formula ϕ *true in* world $\mathcal{M}$?"

Proof theory:

- The structure of formally valid reasoning chains.
- "What kinds of inference steps *preserve truth*?"

Computability theory:

- The things that can (and can not) be programmed.
- *At least:* "Here is a suggested formal proof. Is it valid, or does it contain an invalid step somewhere?"
  - Proof checking.
- *Hopefully also:* "Here is a statement in the logic. Can you find a proof for it?

  Or a counterexample?"
  - Proof search (a.k.a. automatic theorem proving).
  - Counterexample = model where statement fails.

Proof Editors

- Third kind of computational aid:

  No proof search: the user is responsible for choosing the proof strategy.

  No invalid steps: the computer is responsible for preventing mistakes.

- Fourth kind: *Semi-automatic* proof search:
  - The computer asks the user about the most important choices.
  - The computer makes the other choices independently...
  - ...based on user-programmed tactics and tacticals.

Teaching Proof Theory

- Students should learn basics of proof theory before
  - using semi-automatic tools

    (What to answer when the computer asks?)
  - programming their own tactics and tacticals.

    (What patterns of steps are common in proofs?)

- Proof editors are good *educational* tools:

  the student is responsible for planning the whole proof

  the computer is responsible for keeping track of the tiresome details.

The Trinity of Proof Theories

Natural deduction: The most common one in Mathematics and Philosophy courses.

1. Assume ϕ.
2. Prove ψ using assumption 1.
3. *Discharge assumption 1* by concluding "**if** ϕ **then** ψ" from 2.

+ Easy to prove results *about* the logic — few rules.

– Hard to prove results *in* the logic — need skill to come up with good assumptions ϕ.

– Keeping track of open assumptions far away is difficult in a proof editor.

Resolution-like systems: The one explained in depth in Artificial Intelligence courses.

+ Just one, very powerful, machine oriented rule.

+ Efficient in automatic proof search.

– The input formulæ are preprocessed into a uniform format which is hard to read.

– This makes the proofs hard to
  - read
  - guide.

Sequent (a.k.a. Gentzen) systems: The choice for proof editors and semi-automatic proof search.

- Also used in e.g. semantics for natural and programming languages.
- Proof steps manipulate *sequents*:

  all assumed formulæ $\Gamma \Rightarrow$ formula(e) ϕ to infer.

  (Symbol '$\vdash$' is also common.)
- Reading: "Given proofs for elements of Γ, we can construct a proof for ϕ."

- Rules combine sequents:

$$\frac{\text{assumed sequent 1} \quad \text{assumed sequent 2}}{\text{inferred sequent}}\ \text{rule name}$$

- Sequent proof example: transitivity of implication.

$$\dfrac{\dfrac{\dfrac{}{B \supset C, A \Rightarrow A}\,Ax \quad \dfrac{\dfrac{}{B, A \Rightarrow B}\,Ax \quad \dfrac{}{B, C, A \Rightarrow C}\,Ax}{B, B \supset C, A \Rightarrow C}\,L\supset}{A \supset B, B \supset C, A \Rightarrow C}\,L\supset}{(A \supset B)\&(B \supset C), A \Rightarrow C}\,L\&$$
$$\dfrac{(A \supset B)\&(B \supset C), A \Rightarrow C}{(A \supset B)\&(B \supset C) \Rightarrow A \supset C}\,R\supset$$
$$\dfrac{(A \supset B)\&(B \supset C) \Rightarrow A \supset C}{\Rightarrow ((A \supset B)\&(B \supset C)) \supset (A \supset C)}\,R\supset$$

\+ Formulæ retain their original form.

\+ All assumed formulæ Γ remain conveniently grouped together.

– Carrying all assumed formulæ Γ in every step causes lots of redundant bookkeeping.

– Many different rules:

- for each connective (such as 'and' / '&' or 'implies' / '$\supset$')
- on both the Left and Right side of '$\Rightarrow$'.

The proof editor takes care of these problems!

Proof Trees

Nodes are labelled with sequents.

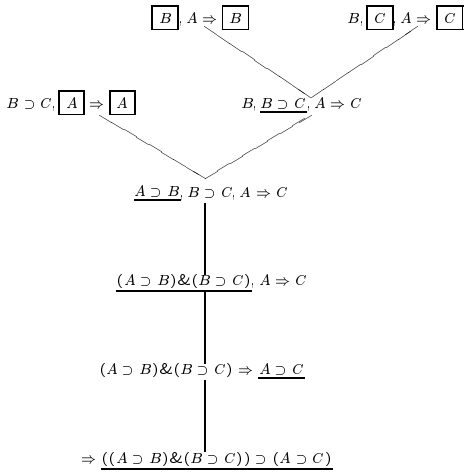
Branching principle:

1. Pick some *node* and some formula in *its* sequent.
2. Apply a **rule** to it.
3. The sequents assumed in **it** are the children.

Leaves come from

rules with no assumed sequents

axioms whose formula in right is in left too.



Proving in Root-First Order

- Proof trees suggest *proving as growing a tree:*
  1. Write the sequent to prove as the root.
  2. Pick any leaf *node*.
  3. Choose a formula and a **rule** *it*.
  4. Add the chosen branches as *its* children.
- The proof is done when all leaves are axioms.
- *This proof attempt* ***fail****s* if we get instead a non-axiom leaf without any possible branching choices.

User Concepts for Picking a *Node*

From the list of current leaves:

+ Easy to implement and use.
– How can we *undo* some previous choice?
  - We must be able to choose also nodes *inside* the tree...
  - ...and delete their current branches.

Spelling out the path from the root explicitly:

+ Easy to implement and undo.
– Cumbersome to use.

Pointing and Clicking:

- Show the proof tree on screen.
- Let the user point to any node with the mouse...
- ...and click on the one he wants.

– More difficult to implement.
+ Easy to use and undo.
+ Global view to the proof is a pedagogically good:
  - The student sees the overall "strategy" of the whole proof.
  - The screen looks like the book.

Picking a formula and rule in the *node*

- The same mouse click can also pick the formula in the *node*.
- It remains to choose the **rule**.
- Choosing **it** should be easy — with the same mouse click if possible.
- Choices for **it** depend on the logic.
- We select a mouse click friendly logic!

Two Kinds of Sequent Calculi

Shared contexts: All branches receive all those assumed formulæ that did not change.

\+ Easy in a point-and-click interface.

Independent contexts: The currently inactive assumed formulæ are divided among the branches in some way.

– Hard in a point-and-click interface.

The Sequent Calculus G3ip

Propositional — no variables ($x_0, x_1, x_2, \ldots$) or quantifiers ($\forall$, $\exists$).

Shared contexts.

Intuitionistic logic:

classical	constructive
true	provable
false	contradictory
law of excluded middle	no proof either way

"Is there life on Mars or not?"

Classical logic is similar — the same user interface applies.

Axioms $P, \Gamma \Rightarrow P$ where

- P is *atomic*
- Γ is a *multiset*.

Rules for 'and', 'or', 'implies' and 'contradiction':

op	left rule	right rule(s)
$\&$	$\dfrac{A, B, \Gamma \Rightarrow C}{A\&B, \Gamma \Rightarrow C}\, L\&$	$\dfrac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A\&B}\, R\&$
$\vee$	$\dfrac{A, \Gamma \Rightarrow C \quad B, \Gamma \Rightarrow C}{A \vee B, \Gamma \Rightarrow C}\, L\vee$	$\dfrac{\Gamma \Rightarrow A}{\Gamma \Rightarrow A \vee B}\, R\vee_1 \quad \dfrac{\Gamma \Rightarrow B}{\Gamma \Rightarrow A \vee B}\, R\vee_2$
$\supset$	$\dfrac{A \supset B, \Gamma \Rightarrow A \quad B, \Gamma \Rightarrow C}{A \supset B, \Gamma \Rightarrow C}\, L\supset$	$\dfrac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A \supset B}\, R\supset$
$\bot$	$\dfrac{}{\bot, \Gamma \Rightarrow C}\, L\bot$	no such rule

Repeating $A \supset B$ in $L\supset$ is omitted in the user interface.

- The **rule** is uniquely determined by
  - the side (left/right) of the formula
  - its outermost `connective`

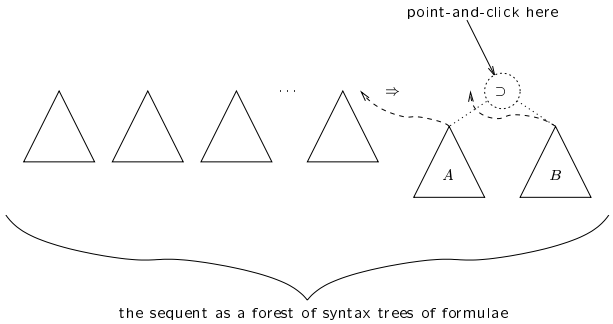
  except for $R\vee_{1/2}$.

- So pointing-and-clicking the formula chooses the **rule** too!

- The **rule**

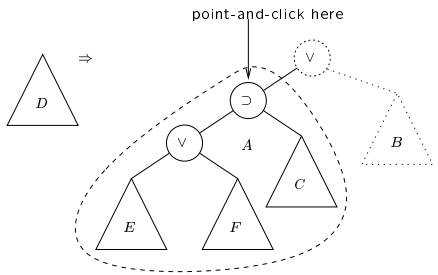
  deletes the `connective`

  divides `its` subformulæ A and B in the (otherwise unmodified) branches.



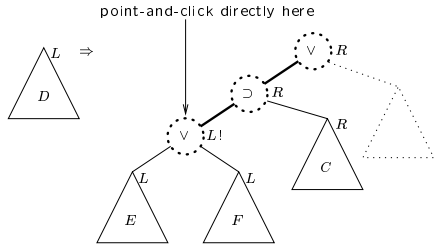
Pointing-and-Clicking Subformulæ

- Pointing-and-clicking can be extended to $R\vee_{1/2}$ by clicking either of its *sub*formulæ A or B.



- Hence *pointing-and-clicking is the only user interface concept* that we need!

- Pointing-and-clicking can be generalized to *all* subformulae:



1. Walk the path from the syntax tree root to the click.
2. Delete syntax tree nodes as you go using the **rule**.
3. Keep track on the side (Left/Right) of the detached subtrees.

- The path
  - gets shorter
  - remains unique

  in the **rule**s.

  We do not have to follow the repeated $A \supset B$ in $L\supset$.

- In this way one click can get you far:

$$\dfrac{\dfrac{\dfrac{D, E \overset{\vdots}{\Rightarrow} C \quad D, F \overset{\vdots}{\Rightarrow} C}{D, E \vee F \Rightarrow C} L\vee}{D \Rightarrow (E \vee F) \supset C} R\supset}{D \Rightarrow ((E \boxed{\vee} F) \supset C) \vee B} R\vee_1$$

How Does it Feel?

- The user points-and-clicks
  1. some part of a formula
  2. inside some sequent of the currently displayed proof.
- Sequent 2 points at *what* sequent the user wants to prove next.

  Or differently, if proving sequent 2 has already begun.
- Part 1 points at *how* the user wants to start proving sequent 2.

  The aim becomes to operate on this part next.

- The system follows the user's aim:
  1. Connectives enclosing the part are deleted using the **rule**s.
  2. Once the part is its own detached element in its sequent, then it can be operated on:

     Deleted as well if it is a connective with a *unique* rule that applies.

     Exposed for further use if it is
     - atomic — the system can (and does) check if its sequent contains a match on the other side of '$\Rightarrow$'
     - Right '$\vee$' — then the user tells which subformula A or B to use with another point-and-click.

       The user can also click A or B directly.

How General Is It?

Assuming that...

- the logic has *shared contexts*
- the rules of the logic *do not make multiple copies* of detached subformulæ
- the series of detaching rules to apply *follow from the path* in the syntax tree

we have arrived at a suitable user interface principle where...

- pointing-and-clicking is enough for everything
- several proof steps can be performed with just one point-and-click.