5th Asian-Pacific Summer School on Formal Methods

August 5-10, 2013, Tsinghua University, Beijing, China

Introduction to trees and proofs

Jean-François Monin

2013-08-05

http://sts.thss.tsinghua.edu.cn/Coqschool2013

June 4, 1996: Ariane 5, flight 501

June 4, 1996: Ariane 5, flight 501

June 4, 1996: firework 501

(Yet another) software problem

Many other examples

Outline

Formal methods and Coq

Trees

Decomposing, case analysis

Functions and implication

Outline

Formal methods and Coq

Trees

Decomposing, case analysis

Functions and implication

Formal Methods

- Prove that some piece of software behaves accordingly to a given specification
- Boils down to theorem proving: programs and specifications are represented by logical formulas
- Hand waving not allowed
- Several complementary approaches and tools

In summary

- Describe a model
- Explain it
- Reason about it
- Be clean and precise

Use math and logic

Some industrial uses

Spacecrafts, airplanes (Airbus, Boeing)

Microsoft
Intel
French railways
Telecom operators
Nuclear power plants
Banks
Cryptography

Coq

A language

- Logic formulas
- Proofs
- Programs

A software proof assistant

Very secure by architectural design

Outline

Formal models, proofs and programs

Based on a common structure: **trees**

- Can be implemented in many ways by software (pointers, arrays, ...)
- This talk: use an intuitive and graphical presentation of trees

Developed by the same activity

which can be

- explicit: using an appropriate syntax for trees
- interactive: using commands for building trees step by step

Make a strong use of **types**

- Everything has a type, even types have a type
- Computations can be carried out on types as well

Basic building blocks for trees: **rules**

$$\frac{\mathit{input}_1 \quad \mathit{input}_2 \quad \ldots \quad \mathit{input}_n}{\mathit{output}}\ \mathit{howto}$$

$\mathit{input}_1, \mathit{input}_2 \ldots, \mathit{input}_n$ and *output* are **types**

howto explains how to make an *output*,
given $\mathit{input}_1, \mathit{input}_2 \ldots, \mathit{input}_n$

Proofs

Same constructs with another reading

Basic building blocks for proof trees: **rules**

$$\frac{\mathit{input}_1 \quad \mathit{input}_2 \quad \ldots \quad \mathit{input}_n}{\mathit{output}}\ \mathit{howto}$$

$\mathit{input}_1, \mathit{input}_2 \ldots, \mathit{input}_n$ and *output* are **propositions**

$\mathit{input}_1, \mathit{input}_2 \ldots, \mathit{input}_n$ are **hypotheses**
output is the **conclusion**

howto justifies how to make a proof of *output*,
given proofs of $\mathit{input}_1, \mathit{input}_2 \ldots, \mathit{input}_n$

Combining rules

Basic building blocks can be combined

Just plug outputs to identical inputs

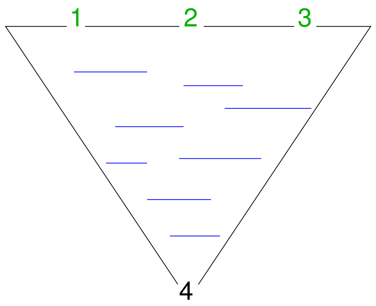
Some rules:

- 1 input comes from exactly 1 output
- an output can be plugged (used) 1, several or 0 times

Alltogether

We get 1 output
from many inputs
using a complex (composed) howto

General shape: trees

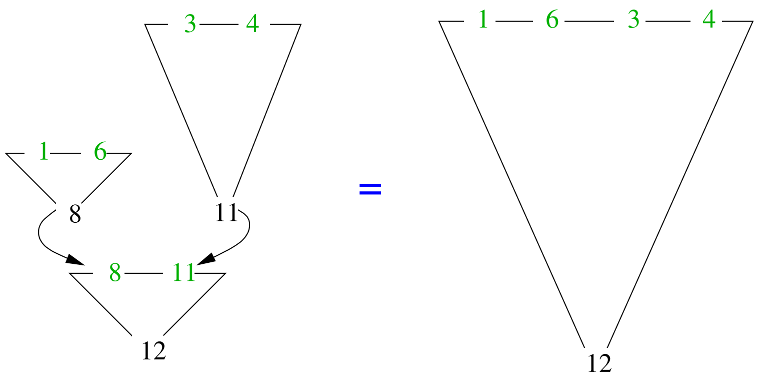


Interpretation

- At positions 1, 2, 3, 4: types
- 1, 2, 3: inputs
- 4: output (or result)

Makes the output from the inputs

Plugging trees



Subtrees can have local additional inputs

Simple examples with 0 input

$\dfrac{}{\texttt{day}}\,\texttt{monday}$ $\dfrac{}{\texttt{day}}\,\texttt{tuesday}$ $\dfrac{}{\texttt{day}}\,\texttt{wednesday}$ $\dfrac{}{\texttt{day}}\,\texttt{thursday}$

$\dfrac{}{\texttt{day}}\,\texttt{friday}$ $\dfrac{}{\texttt{day}}\,\texttt{saturday}$ $\dfrac{}{\texttt{day}}\,\texttt{sunday}$

$\dfrac{}{\texttt{color}}\,\texttt{white}$ $\dfrac{}{\texttt{color}}\,\texttt{black}$

The horizontal bar means: MAKES

Intermezzo: definitions

```
Definition Mo := monday.  Definition Th := thursday.
Definition Tu := tuesday.   Definition Fr := friday.
Definition We := wednesday.  Definition Sa := saturday.
Definition Su := sunday.
```

$\dfrac{}{\texttt{day}}\,\texttt{Mo}$ $\dfrac{}{\texttt{day}}\,\texttt{Tu}$ $\dfrac{}{\texttt{day}}\,\texttt{We}$ $\dfrac{}{\texttt{day}}\,\texttt{Th}$ $\dfrac{}{\texttt{day}}\,\texttt{Fr}$

$\dfrac{}{\texttt{day}}\,\texttt{Sa}$ $\dfrac{}{\texttt{day}}\,\texttt{Su}$

These trees are considered the same as the previous ones
(top of previous slide)

Simple example with 4 identical inputs

Making a 4-tuple of days

$$\dfrac{\texttt{day}\quad\texttt{day}\quad\texttt{day}\quad\texttt{day}}{\texttt{tuple4}}\,\texttt{Mk4day}$$

`Mk4day` makes a tuple4 from

- a day
- a day
- a day
- a day

Plugging day into Mk4day

Building blocks

$\dfrac{}{\texttt{day}}\,\texttt{Mo}$ $\dfrac{}{\texttt{day}}\,\texttt{We}$ $\dfrac{}{\texttt{day}}\,\texttt{Fr}$ $\dfrac{}{\texttt{day}}\,\texttt{We}$

$$\dfrac{\texttt{day}\qquad\texttt{day}\qquad\texttt{day}\qquad\texttt{day}}{\texttt{tuple4}}\,\texttt{Mk4day}$$

Connecting them yields the concrete 4-tuple of day

$$\dfrac{\dfrac{}{\texttt{day}}\,\texttt{Mo}\quad\dfrac{}{\texttt{day}}\,\texttt{We}\quad\dfrac{}{\texttt{day}}\,\texttt{Fr}\quad\dfrac{}{\texttt{day}}\,\texttt{We}}{\texttt{tuple4}}\,\texttt{Mk4day}$$

Another view on Mk4day

As a rule

$$\frac{\texttt{day}\qquad\texttt{day}\qquad\texttt{day}\qquad\texttt{day}}{\texttt{tuple4}}\;\texttt{Mk4day}$$

As a tree

$$\frac{\overbrace{\texttt{day}}^{x_1}\qquad\overbrace{\texttt{day}}^{x_2}\qquad\overbrace{\texttt{day}}^{x_3}\qquad\overbrace{\texttt{day}}^{x_4}}{\texttt{tuple4}}\;\texttt{Mk4day}$$

This is called an open tree

Closed and open trees

The meaning (or value) of

$$\frac{\dfrac{}{\texttt{day}}\,\texttt{Mo}\qquad\dfrac{}{\texttt{day}}\,\texttt{We}\qquad\dfrac{}{\texttt{day}}\,\texttt{Mo}\qquad\dfrac{}{\texttt{day}}\,\texttt{Th}}{\texttt{tuple4}}\;\texttt{Mk4day}$$

is completely defined: this is called a closed tree.

In contrast, the meaning of the open tree

$$\frac{\dfrac{}{\texttt{day}}\,\texttt{Mo}\qquad\overbrace{\texttt{day}}^{x_2}\qquad\dfrac{}{\texttt{day}}\,\texttt{We}\qquad\overbrace{\texttt{day}}^{x_4}}{\texttt{tuple4}}\;\texttt{Mk4day}$$

depends on x_2 and x_4.

Environment

The meaning of the open tree

$$\frac{\dfrac{}{\texttt{day}}\,\texttt{Mo}\qquad\overbrace{\texttt{day}}^{x_2}\qquad\dfrac{}{\texttt{day}}\,\texttt{We}\qquad\overbrace{\texttt{day}}^{x_4}}{\texttt{tuple4}}\;\texttt{Mk4day}$$

depends on x_2 and x_4. It has a meaning for all trees plugged into x_2 and x_4.

The variables x_2 : day and x_4 : day make up the environment of this tree

More 4-tuples

$$\frac{\overbrace{\texttt{day}}^{x_1}\quad\overbrace{\texttt{day}}^{x_2}\quad\overbrace{\texttt{day}}^{x_3}\quad\overbrace{\texttt{day}}^{x_4}}{\texttt{tuple4}}\;\texttt{Mk4day}$$

$$\frac{\overbrace{\texttt{color}}^{x_1}\quad\overbrace{\texttt{color}}^{x_2}\quad\overbrace{\texttt{color}}^{x_3}\quad\overbrace{\texttt{color}}^{x_4}}{\texttt{tuple4}}\;\texttt{Mk4co}$$

$$\frac{\overbrace{\texttt{tuple4}}^{x_1}\quad\overbrace{\texttt{tuple4}}^{x_2}\quad\overbrace{\texttt{tuple4}}^{x_3}\quad\overbrace{\texttt{tuple4}}^{x_4}}{\texttt{tuple4}}\;\texttt{Mk4t4}$$

Lists of days

$$\frac{}{\texttt{daylist}}\ \texttt{nil}$$

$$\frac{\overbrace{\texttt{day}}^{d}\quad\overbrace{\texttt{daylist}}^{u}}{\texttt{daylist}}\ \texttt{cons}$$

Examples with proofs

Consider basic propositions, e.g.:

Proposition	**Intended meaning**
H	hot – the temperature is greater than 35 C
W	the glass contains water
C	the blackboard is clean
HaW	the temperature is greater than 35 C and the glass contains water
HoW	the temperature is greater than 35 C or the glass contains water

Given a proof h of H and a proof w of W,
what is a proof of HaW?
what is a proof of HoW?

Examples with proofs

$$\frac{\overbrace{\texttt{H}}^{h}\quad\overbrace{\texttt{W}}^{w}}{\texttt{HaW}}\ \texttt{conj}_{\texttt{HW}}$$

$$\frac{\overbrace{\texttt{H}}^{h}}{\texttt{HoW}}\ \texttt{case1}_{\texttt{HoW}}\qquad\qquad\frac{\overbrace{\texttt{W}}^{w}}{\texttt{HoW}}\ \texttt{case2}_{\texttt{HoW}}$$

To go farther: making propositions

Given 2 propositions P and Q, make a new proposition whose intended meaning is:
P and Q hold together.

This proposition is noted $P \wedge Q$,
Note that it is itself a tree (at the level of propositions, not at the level of proofs)

$$\frac{\overbrace{\texttt{Prop}}^{P}\quad\overbrace{\texttt{Prop}}^{Q}}{\texttt{Prop}}\ \texttt{and}$$

Similarly, $P \vee Q$ represents the tree:

$$\frac{\overbrace{\texttt{Prop}}^{P}\quad\overbrace{\texttt{Prop}}^{Q}}{\texttt{Prop}}\ \texttt{or}$$

General conjunctions and disjunctions

$$\frac{\overbrace{\texttt{P}}^{p} \qquad \overbrace{\texttt{Q}}^{q}}{\texttt{P} \land \texttt{Q}}\ \texttt{conj}$$

$$\frac{\overbrace{\texttt{P}}^{p}}{\texttt{P} \lor \texttt{Q}}\ \texttt{or\_left} \qquad\qquad \frac{\overbrace{\texttt{Q}}^{q}}{\texttt{P} \lor \texttt{Q}}\ \texttt{or\_right}$$

Outline

Decomposing, case analysis

- Given a 4-tuple t, extract its components
- Given a day d, provide a color depending on d
- Given a color c, provide a day depending on c
- Given a proof of $A \land B$, provide a proof of A
- Given a proof of $A \lor B$, provide a proof of $B \lor A$
  A proof of $B \lor A$ is needed in each case

Case analysis

Question

Give a `day` for each possible value `c` in `color`

$$\frac{}{\texttt{color}}\ \texttt{white} \qquad \frac{}{\texttt{color}}\ \texttt{black}$$

Example

`white` maps to `thursday`, `black` maps to `monday`

$$\frac{\genfrac{}{}{0pt}{}{\vdots\ \texttt{c}}{\texttt{color}} \qquad \frac{}{\texttt{day}}\ \texttt{thursday} \qquad \frac{}{\texttt{day}}\ \texttt{monday}}{\texttt{day}}\ \texttt{destruct}$$

Warning: the order of branches matters

Does it make sense? Subtle point!

Statement of the previous question

Give a `day` for each possible value in `color`

Here assume that all trees with `color` as output are (in this order)

$$\text{either} \quad \frac{}{\texttt{color}}\ \texttt{white} \qquad\qquad \text{or} \quad \frac{}{\texttt{color}}\ \texttt{black}$$

The real story is more subtle

- Claim of exhaustivity: related to inductive types
- However, there are (infinitely) many trees which make a `color`
- However, they eventually reduce to one of the declared cases: related to computations and so-called *strong normalization*

Building block of a case analysis

In this presentation, the order of contructors matters:
`white`, `black`
The `destruct` construct is driven by 2 parameters

- the type of the value to be analyzed
  each enumerated type (e.g., `color`) comes automatically with its `destruct` construct, which should actually be written, e.g. $\texttt{destruct}_{\texttt{color}}$
- the type of the result (e.g., `day`)

$$\frac{\overbrace{\texttt{Set}}^{A} \quad \overbrace{\texttt{color}}^{c} \quad \overbrace{\texttt{A}}^{x_2} \quad \overbrace{\texttt{A}}^{x_3}}{\texttt{A}}\ \texttt{destruct}$$

Correct version of the previous example

$$\frac{\dfrac{}{\texttt{Set}}\ \texttt{day} \quad \overbrace{\texttt{color}}^{c} \quad \dfrac{}{\texttt{day}}\ \texttt{thursday} \quad \dfrac{}{\texttt{day}}\ \texttt{monday}}{\texttt{day}}\ \texttt{destruct}$$

Decomposing

Original question

Given a proof tree p of $\texttt{A} \wedge \texttt{B}$, provide a proof of `A`
We know that the only possible shape of p is

$$\frac{\begin{matrix} \vdots\ \texttt{a} \\ \texttt{A} \end{matrix} \qquad \begin{matrix} \vdots\ \texttt{b} \\ \texttt{B} \end{matrix}}{\texttt{A} \wedge \texttt{B}}\ \texttt{conj}$$

where $\boxed{\begin{matrix} \vdots\ \texttt{a} \\ \texttt{A} \end{matrix}}$ is an arbitrarily large proof tree concluding to `A`

and similarly for $\boxed{\begin{matrix} \vdots\ \texttt{b} \\ \texttt{B} \end{matrix}}$

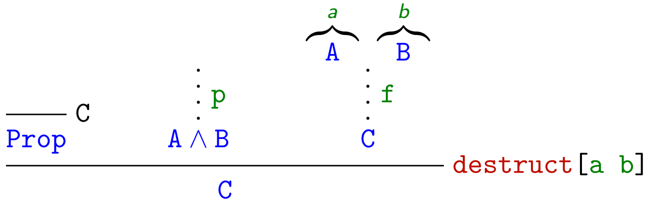
So the answer is `a`, but how to get it?

Decomposing

More general question

Given a proof tree p of A ∧ B,
prove some proposition C using the two components building *p*.

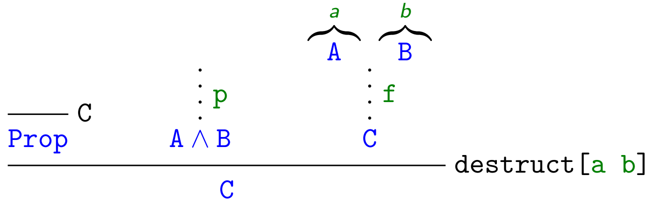
Rule

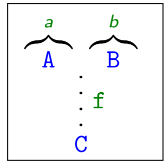


Reading

Let us prove C assuming *a*, a proof of A and *b*, a proof of B; as we have a proof of A ∧ B, we get a proof of C.

Warning

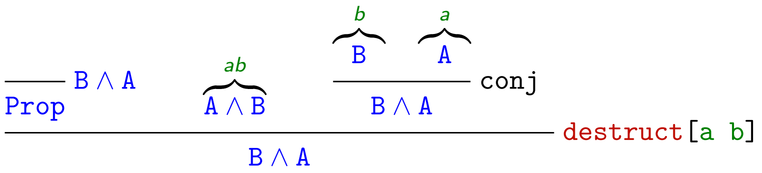


The subtree 

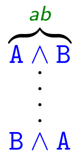
contains additional local inputs *a* and *b*

a and *b* are not available outside f

Example: a proof of B ∧ A from A ∧ B

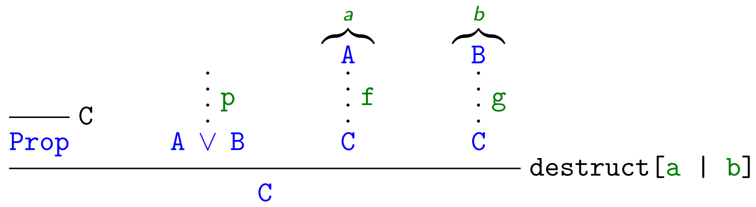


Shape of this proof tree



Case analysis and decomposition

For a disjunction A ∨ B, we have 2 cases, and each one has 1 input



The hypothesis *a* is available only inside f
The hypothesis *b* is available only inside g

Exercises

- a proof of B ∨ A from A ∨ B
- a proof of A from A ∧ B
- a proof of B ∧ B from A ∧ B
- draw the case analysis and decomposition rules for 4-tuples

Outline

Formal methods and Coq

Trees

Decomposing, case analysis

Functions and implication

Implication

What is a function from day to color?

- open a new scope where d represents an arbitrary tree with day as output
- make a tree with output color from d
- in this subtree, d is available; but not outside

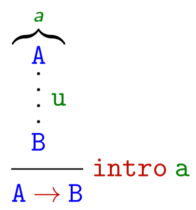
What is a proof that P implies Q?

- open a new scope with p an arbitrary proof of P (an arbitrary proof tree with P as the conclusion)
- make a proof tree with conclusion Q from p
- in this subtree, p is available; but not outside

It is just a function from P to Q

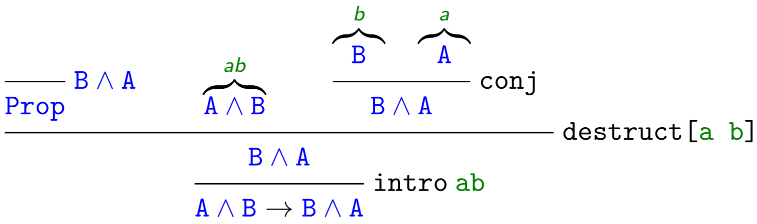
Again: some subtrees have additional inputs

Introduction rules for implications/functions

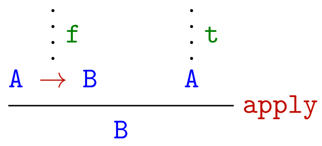


Warning: this a is available only in u

Example



Applying a function / modus ponens



Next lectures and classes

Use Coq to build
formal models, programs and proofs
(trees)

using an explicit syntax or in the interactive way