

Phometa — a visualised proof assistant that builds a formal system and proves its theorems using derivation trees

Gun Pinyo

Imperial College London

June 21, 2016

Derivation systems can reason many formal systems.

$$\text{IMPLY-ELIM} \frac{\Gamma \vdash (A \rightarrow B) \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\frac{\frac{\frac{}{p, p \rightarrow q \vdash p} \text{ ASS} \quad \frac{\frac{}{p, p \rightarrow q \vdash p \rightarrow q} \text{ ASS} \quad \frac{}{p, p \rightarrow q \vdash p} \text{ ASS}}{p, p \rightarrow q \vdash q} \text{ IMPLY-ELIM}}{p, p \rightarrow q \vdash p \wedge q} \text{ AND-INTRO}$$

Derivation systems can reason many formal systems.

$$\begin{array}{c}
 \frac{}{(w \times (x + y)) = ((w \times x) + (w \times y))} \text{DIST-LEFT} \\
 \frac{}{((w \times x) + (w \times y)) = (w \times (x + y))} \text{EQ-SYMM} \quad \frac{}{z = z} \text{EQ-REFL} \\
 \hline
 (((w \times x) + (w \times y)) \times z) = ((w \times (x + y)) \times z) \quad \text{MULT-INTRO}
 \end{array}$$

$$\begin{array}{c}
 \frac{}{y : A, x : A \vdash x : A} \text{ASSUMPTION} \\
 \frac{}{y : A \vdash \lambda x. x : A \rightarrow A} \text{ARROW-INTRO} \quad \frac{}{y : A \vdash y : A} \text{ASSUMPTION} \\
 \hline
 y : A \vdash (\lambda x. x)y : A \quad \text{ARROW-ELIM}
 \end{array}$$

Problem of drawing a derivation tree.

- Its width grows exponentially to its height.
- One drawing mistake might need major correction of the entire tree.
- (Meta) is being rewritten by unification.
- Cannot systematically check that the tree has no errors.

Screenshot of “Phometa” in action

The screenshot displays the Phometa software interface, which is used for defining and proving theorems in a formal logic system. The interface is divided into several panels:

- Repository Panel (Left):** Shows a tree view of the library structure. The 'Standard Library' is expanded, showing 'Propositional Logic' and 'demo-theorem-1'. Below this, there are buttons for 'Add Package' and 'Add Module'.
- Main Editor Panel (Center):** Displays the content of the 'demo-theorem-1' module. It shows a theorem definition and its proof steps.

Theorem `demo-theorem-1` = $\epsilon, p, p \rightarrow q \vdash p \wedge q$

The proof is structured as follows:

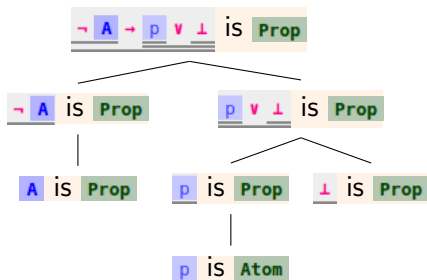
 - Step 1: $\epsilon, p, p \rightarrow q \vdash p$ (proof_by_rule hypothesis)
 - Step 2: $\epsilon, p, p \rightarrow q \vdash p \rightarrow q$ (proof_by_rule hypothesis)
 - Step 3: $\epsilon, p, p \rightarrow q \vdash p$ (proof_by_rule hypothesis)
 - Step 4: $\epsilon, p, p \rightarrow q \vdash q$ (proof_by_rule imply-elim with $A = p$)
 - Step 5: $\epsilon, p, p \rightarrow q \vdash p \wedge q$ (proof_by_rule and-intro)
- Key Panel (Right):** Provides a list of keyboard shortcuts and their descriptions.

Key	Description
Alt-l	lock as lemma
Alt-r	reset whole theorem
Alt-t	reset current proof
Alt-x	jump to menu

At the bottom of the main editor, there is a **Comment** section with the text: '// Module Description'. Below this, a paragraph states: 'This module implements a formal system named "Propositional Logic" which is the most well known logical system.'

Formal system ingredient - Grammar (Backus-Nour Form)

```
<Prop> ::= 'T' | '⊥' | <Atom>
         | '(' <Prop> '^' <Prop> ')'
         | '(' <Prop> '∨' <Prop> ')'
         | '(' ¬ <Prop> ')'
         | '(' <Prop> '→' <Prop> ')'
         | '(' <Prop> '↔' <Prop> ')'
         | meta-variable with regex
         / [A-Z][a-zA-Z]*([1-9][0-9]*|'*)/
```



```
<Atom> ::= literal with regex
         / [a-z][a-zA-Z]*([1-9][0-9]*|'*)/
```

Grammar Atom ✕

literal_regex [a-z][a-zA-Z]*([1-9][0-9]*|'*)

Grammar Prop ✕

metavar_regex [A-Z][a-zA-Z]*([1-9][0-9]*|'*)

choice T

choice ⊥

choice Atom

choice Prop ^ Prop

choice Prop ∨ Prop

choice ¬ Prop

choice Prop → Prop

choice Prop ↔ Prop

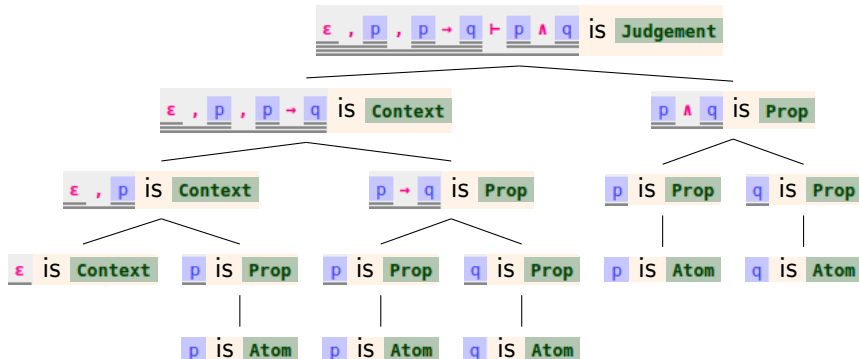
Grammar (Backus-Nour Form)

```
<Context> ::= 'ε'
           | <Context> ',' <Prop>
           | meta-variables comply with regex
             / [ΓΔ] ([1-9] [0-9]* | '*' ) /
```

```
<Judgement> ::= <Context> '⊢' <Prop>
```

Grammar	Context	<input type="button" value="X"/>
metavar_regex	[ΓΔ]([1-9][0-9]* '*)	
choice	ε	
choice	Context	Prop

Grammar	Judgement	<input type="button" value="X"/>
choice	Context	Prop



Formal system ingredient - Rule (Derivation Rule)

$$\text{IMPLY-ELIM} \frac{\Gamma \vdash (A \rightarrow B) \quad \Gamma \vdash A}{\Gamma \vdash B}$$

Rule `imply-elim`

premise $\Gamma \vdash A \rightarrow B$

premise $\Gamma \vdash A$

conclusion $\Gamma \vdash B$

parameter $A : \text{Prop}$

$\varepsilon, p, p \rightarrow q \vdash p \rightarrow q$
to_prove
Proof By Rule

$\varepsilon, p, p \rightarrow q \vdash p$
to_prove
Proof By Rule

$\varepsilon, p, p \rightarrow q \vdash q$
proof_by_rule `imply-elim` with $A = p$

Pattern Matching

$$\Gamma = \varepsilon, p, p \rightarrow q$$

$$B = q$$

Parameter(s)

$$A = p$$

Formal system ingredient - Theorem (Derivation Tree)

$$\begin{array}{c}
 \frac{}{p, p \rightarrow q \vdash p} \text{ ASS} \quad \frac{}{p, p \rightarrow q \vdash p} \text{ ASS} \\
 \frac{}{p, p \rightarrow q \vdash p} \text{ ASS} \quad \frac{p, p \rightarrow q \vdash p \rightarrow q \quad p, p \rightarrow q \vdash p}{p, p \rightarrow q \vdash p} \text{ IMPLY-ELIM} \\
 \hline
 p, p \rightarrow q \vdash p \wedge q \quad \text{AND-INTRO}
 \end{array}$$

Theorem `demo-theorem-1` = $\varepsilon, p, p \rightarrow q \vdash p \wedge q$

$\varepsilon, p, p \rightarrow q \vdash p$ **proof_by_rule** `hypothesis`

$\varepsilon, p, p \rightarrow q \vdash p \rightarrow q$ **proof_by_rule** `hypothesis`

$\varepsilon, p, p \rightarrow q \vdash p$ **proof_by_rule** `hypothesis`

$\varepsilon, p, p \rightarrow q \vdash q$ **proof_by_rule** `imply-elim` with **A** = `p`

$\varepsilon, p, p \rightarrow q \vdash p \wedge q$ **proof_by_rule** `and-intro`

Demonstration

- Overview of Phometa's user interface.
- Prove that $\varepsilon, p, p \rightarrow q \vdash p \wedge q$ is a valid **Judgement**.
- Extend Propositional Logic to support equivalence.

<Equivalence> ::= **<Prop>** '≡' **<Prop>**

$$\text{EQUIV-INTRO} \frac{A \vdash B \quad B \vdash A}{A \equiv B}$$

- Prove that $\neg P \vee Q \equiv \neg P \wedge \neg Q$ is a valid **Equivalence**.

Cascade Premise — Hypothesis Rule

Rule **hypothesis** ✕

cascade

- hypothesis-base** **exact_match** $\Gamma, B \vdash A$
- hypothesis** $\Gamma \vdash A$

conclusion $\Gamma, B \vdash A$

Rule **hypothesis-base** ✕

conclusion $\Gamma, A \vdash A$

Meta-Reduction — Context Manipulation

Rule context-commutative ✕

premise Γ, B, A

conclusion Γ, A, B

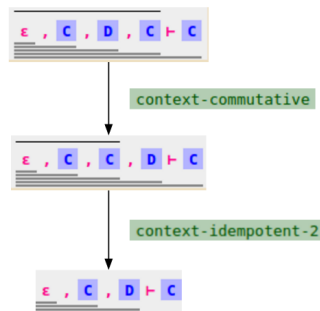
allow_reduction

Rule context-idempotent-2 ✕

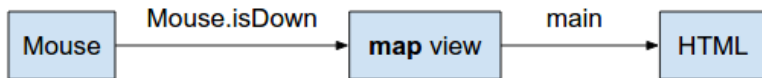
premise Γ, A

conclusion Γ, A, A

allow_reduction



Implementation — Elm language and its Signals



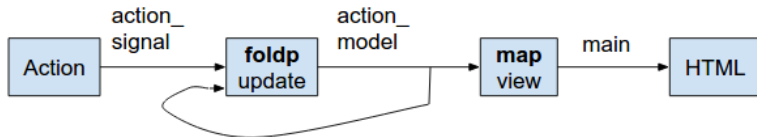
```
Mouse.isDown : Signal Bool
```

```
Signal.map    : (a -> b) -> Signal a -> Signal b
```

```
view : Bool -> Html
view is_clicked =
    if is_clicked then text "bar"
    else text "foo"
```

```
main : Signal Html
main = Signal.map view Mouse.isDown
```

Implementation — Model-Controller-View (MCV)



```
Signal.foldp : (a -> b -> b) -> b -> Signal a -> Signal b
```

```
update : Action -> Model -> Model
```

```
view : Model -> Html
```

```
action_signal : Signal Action
```

```
action_signal = Signal.merge mailbox.signal keyboard_signal
```

```
model_signal : Signal Model
```

```
model_signal = Signal.foldp update init_model action_signal
```

```
main : Signal Html
```

```
main = Signal.map view model_signal
```

Remark: this is the *simplified* version of Phometa main entry.

Strengths, Limitation, and Future Work

- Has extra features over the traditional derivation system such as prove by lemma, cascade premise, and meta-reduction.
 - Less learning curve than mainstream proof assistants.
 - The program always in consistent state
i.e. impossible to create an invalid proof.
-
- Phometa need to load the entire proof repository when start.
 - Theorem building process should be more automatic.
 - Nodes should be able to be exported into $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ source code.
 - Modules should be able to import nodes from other modules.

Achievement & Conclusion

- Finished designing Phometa specification in such a way to keep it simple yet be able to produce a complex proof.
- Finished implementing Phometa, as you have seen in the demo.
- Encoded these formal systems in the standard library.
 - ▶ Propositional Logic
 - ▶ Simple Arithmetic
 - ▶ Typed Lambda Calculus
- Wrote a tutorial for newcomers to use Phometa (Chapters 3, 4, A, B, C in the report).
- Phometa is ready to be used as a replacement of the traditional derivation system.

Questions & Answers

Thank You

