Derivatives and Expression simplification

Learning unit 3 - report

1837592 - Andrea Princic
1852473 - Valerio Venanzi

July 19, 2022

Contents

1 Age

This learning unit is meant for students of 4^{th} year of high school (Liceo Scientifico).

Derivatives are usually studied during the fourth year, so this learning unit can't be done before that.

2 Pre-requisites and Goals

2.1 Interdisciplinary

The goal is to teach students how to solve derivatives. This is done in two ways:

- simple derivatives are solved directly
- more complicated derivatives are solved with more complicated rules, that, in order to be written, required the student to better understand how derivates are solved

The same goes for the simplification of expressions.

Preliminary knowledge of **derivatives** and **arithmetic** is needed to complete the task.

2.2 Programming

The goal is to teach students the basics of expression/syntax manipulation in Prolog.

One of the main points is that students can understand that mathematical expressions can be solved syntactically, and not only by sheer calculations.

3 Problem

Students are asked to write a program that solves two different tasks:

- Compute derivatives
- Simplify expressions

3.1 Compute derivatives

Given a function/expression, the program should be able to calculate its derivative, w.r.t. a variable.

Example:

Let $3 \cdot sin(x + 2) + 6$ be an expression, we calculate its derivative, which is $3 \cdot (cos(x + 2) \cdot (1 + 0)) + 0$ after applying every derivation rules, without any simplification.

3.2 Simplify expressions

Given an expression the program should be able to evaluate it, that is, it brings it to a simpler term.

Example:

$sin(x) + ((10 + 14) \cdot 3) + 3 \cdot tan(x)$ simplifies to $sin(x) + 42 + 3 \cdot tan(x)$.

3.3 Simplify derivatives

The final task is to combine derivation and expression simplification to compute the simplified derivative of a function, w.r.t. the given variable.

The order in which they are meant to be applied is, given an expression E:

1. Simplify E into E_1
2. Derivate E_1 into E_1'
3. Simplify E_1' into E_2

One can choose not to simplify the first time.

Making some complete examples:

Let $E = 3 \cdot sin(x + 2) + 6$ be an expression.

1. simplify it: $simplify(E) = E = 3 \cdot sin(x + 2) + 6$ (this expression doesn't simplify)
2. derivate it: $E' = 3 \cdot (cos(x + 2) \cdot (1 + 0)) + 0$
3. simplify it: $simplify(E') = 3 \cdot cos(x + 2)$

Let $E = x + sin(x) + e^x + log_3(x + 1)$ be an expression.

1. derivate it: $E' = 1 + cos(x) \cdot 1 + e^x + \frac{1}{(x+1) \cdot ln(e)} \cdot (1 + 0)$
2. simplify it: $simplify(E') = 1 + cos(x) + \frac{1}{(x+1) \cdot ln(3)} + e^x$

Let $E = (x^2 + ln(x)) \cdot arcsin(x)$ be an expression.

1. derivate it: $E' = (x^2 + ln(x)) \cdot (\frac{1}{\sqrt{1-x^2} \cdot 1}) + arcsin(x) \cdot (2 \cdot x^{2-1} + \frac{1}{x})$
2. simplify it: $simplify(E') = (x^2 + ln(x)) \cdot (\frac{1}{\sqrt{1-x^2}}) + arcsin(x) \cdot (2 \cdot x + \frac{1}{x})$

4 System: Prolog

The learning unit will be implemented in Prolog.

Although the first impact with Prolog can be a bit harsh, once the students understand its basic functioning, writing a program (in particular the program for this learning unit) should be quite easy.

Prolog makes it really easy to write simple rules that define the rules for derivation and expression simplification; that is, it makes manipulating mathematical expression easy.

Function	Derivative
$f(x) = k$	$f'(x) = 0$
$f(x) = x^n$	$f'(x) = n \cdot x^{n-1}$
$f(x) = \frac{1}{x}$	$f'(x) = -\frac{1}{x^2}$
$f(x) = \sqrt[n]{x}$	$f'(x) = \frac{1}{n\sqrt[n]{x^{n-1}}}$
$f(x) = \lvert x \rvert$	$f'(x) = \frac{\lvert x \rvert}{x}$
$f(x) = log_a(x)$	$f'(x) = \frac{1}{x} log_a(e)$
$f(x) = ln(x)$	$f'(x) = \frac{1}{x}$
$f(x) = a^x$	$f'(x) = a^x ln(a)$
$f(x) = e^x$	$f'(x) = e^x$

Table 1: Simple expressions

5 Learning materials

Students will have access to:

- Prolog's documentation
- A cheat sheet with the derivation rules

Students will also be encouranged to search examples online that can help them.

6 LU delivery

Students will work in pairs (i.e. pair programming).

We assume that the students have already seen Prolog (i.e. we've studied it in class). After giving them the task, they work at home in order to complete it; if they struggle, we can spend a laboratory class on the learning unit.

They have a week of time.

7 Evaluation grid

7.1 Correctness of the solution

We expect a sufficient solution to be able to derivate functions that only need a single syntactic rule application; that is, all the rules in Simple expressions and

Function	**Derivative**
$f(x) = sin(x)$	$f'(x) = cos(x)$
$f(x) = cos(x)$	$f'(x) = -sin(x)$
$f(x) = \ tan(x)$	$f'(x) = \ \frac{1}{cos^2(x)} = 1 + tan^2(x)$
$f(x) = \ cot(x)$	$f'(x) = \ -\frac{1}{sin^2(x)} = -(1 + cot^2(x))$
$f(x) = \ arcsin(x)$	$f'(x) = \ \frac{1}{\sqrt{1-x^2}}$
$f(x) = arccos(x)$	$f'(x) = \ -\frac{1}{\sqrt{1-x^2}}$
$f(x) = \ arctan(x)$	$f'(x) = \ \frac{1}{1+x^2}$
$f(x) = arccot(x)$	$f'(x) = \ -\frac{1}{1+x^2}$

Table 2: Trigonometric functions

Function	**Derivative**
$k \cdot \ f(x)$	$k \cdot f'(x)$
$f(x) + g(x)$	$f'(x) + g'(x)$
$f(x) \cdot g(x)$	$f'(x) \cdot g(x) + f(x) \cdot g'(x)$
$\frac{f(x)}{g(x)}$	$\frac{f'(x) \cdot g(x) - f(x) \cdot g'(x)}{g^2(x)}$
$\frac{1}{f(x)}$	$-\frac{f'(x)}{f^2(x)}$
$f(g(x))$	$f'(g(x)) \cdot g'(x)$

Table 3: Derivation rules

	Sufficient	**Good**	**Outstanding**
Correctness of the solution	Simple derivatives and expressions must be computed correctly	Every derivative and expression is computed correctly	Every derivative and expression is computed correctly
Completeness of the solution	The program works for simple expression	The program work also for more complicated expression	The program works for every expression
Code quality	Naming convention not coherent and uncommented code	Naming convention coherent, not very commented	Naming convention coherent, commented code and short simple rules

Trigonometric functions.

Good and outstanding should be able to work with any given function; this also include Derivation rules.

7.2 Completeness of the solution

This more or less follows correctness, with one caveat.

The solution is correct if it works, that is, the result of derivating or simplifying and expression, is correct. Instead, a solution is complete if the rules have been implemented but some of them may containg bugs; of course rules must not be implemented randomly, but a few bugs (that break the program for complex expressions) are tolerated.

In other words, Correctness requires Completeness and Correctness, while Completeness doesn't require full Correctness.

7.3 Code quality

We expect rules to be:

- short
- simple
- descriptive

Rules that are short and simple (i.e. they do one thing only) will be encouranged.

By *descriptive* we mean that variables' names recall what they represent; e.g. a function should be called F, G, H.

We also encourange to be coherent with naming throughout all the program; e.g. if constants are represent with the first letters of the alphabet (i.e. A, B, C) then this should be done in every rule where a constant appears.

8 Solutions

8.1 Best solution (10/10)

The solution is able to derivate and simplify complex expression using generalized rules.

Predicates are easy to understand, thanks to both comments and clear/simple rules.

It is able to derivate every rule in the tables given; i.e. Simple expressions, Trigonometric functions and Derivation rules. This lets the program derivate every complex expression.

Same goes for simplification.

Naming convention is coherent and descriptive.

8.2 Sufficient solution (6/10)

The program implements only the rules in

- Simple expressions
- Trigonometric functions
- from Derivation rules the rules for sum of two functions and product of two functions

The program is poorly commented.

Naming convention is not followed and variables' names are not very descriptive.