Programming Paradigms

Programming Paradigms	
Richting	Informatica
Jaar	MINF

Examenvragen

Academiejaar 2020 - 2021 - 1ste zittijd

Before starting, make sure you read all the questions and understand what is asked of you. Except for multiple-choice questions, write all your answers in a separate sheet of paper and make sure you justify yes/no answers. (You can make use of combinators or functions defined during class.)

1. (5 points) Which elements from the following list are **not** programming paradigms (that we mentioned during the course)?

- Linear programming
- Logic programming
- Functional programming
- Quadratic programming
- Differentiable programming

Lambda calculus

1. (10 points) Recall that the combination for true and false are λx.λy.xλx.λy.x and λx.λy.yλx.λy.y respectively. Prove that the conditional combinator COND =λe1.λe2.λc.((ce1)e2)=λe1.λe2.λc.((ce1)e2) is correct. That is, prove that (((CONDd1)d2)f)(((CONDd1)d2)f) reduces to d1d1 if ff is ββ-equivalent to true and d2d2 if ff is ββ-equivalent to false.

1. (10 points) If AA is ββ-equivalent to BB and AA has no normal form then:
   1. BB could have a normal form
   2. BB has a normal form
   3. BB had no normal form

1. (10 points) Define a combinator that, given a list of elements LL, returns LL concatenated with LRLR, where LRLR denotes the reversed version of LL.

1. (10 points) Define a **tail-recursive** combinator that, given a list of Boolean elements, returns a Boolean value of true if and only if **all the elements in the list are true.**

1. (15 points) Define a combinator that, given two numbers in Church-numeral encoding, returns their *greatest common divisor.* **You may use −,=,≠\textless,≤−,=,≠\textless,≤ combinators:** SUB, EQ, NE, LT, LE.

1. (10 points) Which of the following reduction strategies guarantees that we reach a normal form if one exists?

- Applicative order: reduce the arguments first
- Normal order: reduce the function definition first

1. (10 points) Argue that the following is a fixpoint combinator ((λx.λy.(y((xx)y)))(λx.λy.(y((xx)y))))

((λx.λy.(y((xx)y)))(λx.λy.(y((xx)y))))

1. (5 points) If we restrict the number of combinators we *use* and *define*, how many would be sufficient to be able to capture the whole λλ-calculus? That is, to define any function definable with the whole λλ-calculus.

Prolog

1. (5 points) In your own words, describe what the difference is between *red and green cuts?*

1. (10 points) Define a Prolog predicate that, given an list LL, returns a list where all **consecutively repeated elements** in LL are replaced by a single copy.

- E.g. **?- compress([a,a,b,c,c,c,a,c],X)**
- yields **X = [a,b,c,a,c]**

Academiejaar 2018 - 2019 - 1ste zittijd

Before starting, make sure you read all the questions and understand what is asked of you. Except for multiple-choice questions, write all your answers in a separate sheet of paper and make sure you justify yes/no answers. (You can make use of combinators or functions defined during class, but make sure to reference the lecture and slide numbers.)

1. Which elements from the following list are not programming paradigms? (5 points)

- Logic programming
- Functional programming
- Linear programming
- Probabilistic programming
- Automata-based programming

- Differentiable programming

1. Bonus! Describe a problem which could be solved using constraint programming, that is to say using SAT or SMT solvers.

Lambda calculus

1. Define a combinator that, given a list of elements LL, returns LL concatenated with LRLR, where LRLR denotes the reversed version of L. (10 points)
2. Define a tail-recursive combinator that, given a list of Boolean elements, returns a Boolean value of true if and only if there exists a true element in the list. (10 points)
3. Define a combinator that, given two numbers in Church-numeral encoding, returns their greatest common divisor. You may use −,=,≠,<and≤−,=,≠,<and≤ combinators: SUB, EQ, NE, LT, LE. (15 points)
4. Argue that the following is a fixpoint combinator. ((λx.λy.(y((xx)y)(λx.λy.(y((xx)y)))) ((λx.λy.(y((xx)y)(λx.λy.(y((xx)y)))) (15 points)
5. Determine the type of the following λλ-expressions given T{a}=T{x}=σ,T{b}=τ,T{g}=(σ→τ)→σT{a}=T{x}=σ,T{b}=τ,T{g}=(σ→τ)→σ.
    1. λg.(g(ab))λg.(g(ab)) (5 points)
    2. (λa.(gλc.b)c)(λa.(gλc.b)c) (5 points)

Are they type correct?

Prolog

1. In your own words, describe what the difference is between red and green cuts? (5 points)
2. Define a Prolog predicate that, given a list L, returns a list where all consecutively repeated elements in L are replaced by a single copy. (10 points)
    1. E.g. ?- compress([a, a, b, c, c, c, a, c], X).
    2. yields X = [a, b, c, a, c]
3. Define a packing predicate in Prolog. (15 points)
    1. Define a Prolog predicate that, given a list L, returns a list where consecutively repeated elements in L have been packed into sublists. (10 points)
        1. E.g. ?- pack([a, a, b, c, c, c, a, c], X).
        2. yields X = [[a, a], [b], [c, c, c], [a], [c]]
    2. Make your predicate flexible so that ?- pack(X, [[a, a], [b], [c, c, c], [a], [c]]) yields X = [a, a, b, c, c, c, a, c]. (5 points)
4. Define a Prolog predicate that, given a list L, returns a compressed list using run-length encoding for packed consecutively repeated elements. That is, modify your packing predicate to output a length-letter pair instead of a list of appearances. (5 points)
    1. E.g. ?- encode([a, a, b, c, c, c, a, c], X).
    2. yields X = [[2, a], [1, b], [3, c], [1, a], [1, c]]