IA165
Combinatory Logic for Computational Semantics

Spring 2012

Juyeon Kang

qkang@fi.muni.cz

B410, Faculty of Informatics, Masaryk University,
Brno, Czech Rep.

Interesting Readings

Curry, Haskell B. **Combinatory Logic.** Vol. 1 by Curry and R. Feys; vol. 2 by Curry, J.R. Hindley, and J.P. Seldin. North-Holland, 1958, 1972.

Fitch, Frederic. **Elements of Combinatory Logic.** Yale University Press, 1974.

Hindley, R., B. Lercher, J. Seldin. **Introduction to Combinatory Logic.** Cambridge University Press, 1972.

Lecture 2

- Introduction to the Combinatory Logic

Historical background on CL

1) first invented by M.I Schönfinkel in 1920s

what for? Elimination of bound variables

example: see the table...

2) abstract operators: combinators

Z, T, I, C, S => B, C, I, K, S

K and S define the other three combinators.

- Main idea

  from **K** ans **S**, with a logical operator, one can generate all formulas of predicate logic without the use of bound variables.

- Extra remark

  multi-variable applications such as F(x,y) can be replaced by

  (f(x))(y) where f is a function whose output-value f(x) is also a function => **Currying**

CL by Haskell Curry

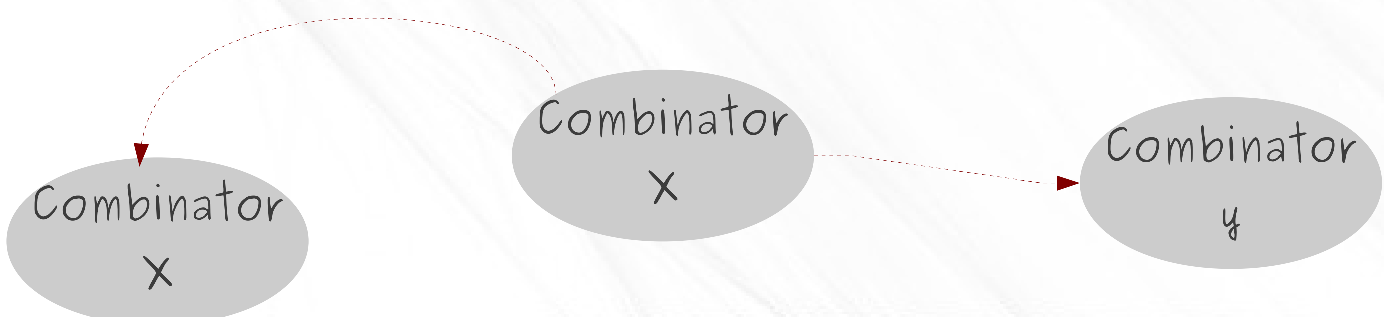
1) a formal system of combinators and a proof of the combiantory completeness of {B, C, K, W}

completeness proof => abstraction algorithm (coming next slides)

- **Important remark 1**

For Curry, as Schönfinkel,

every combinator was allowed to be applied to every other combinator and even to itself.



- <u>Important remark 2</u>

All expressions of CL are applicative expressions where an operator is applied to an operand. CL is generated from abstract operators, called combinators, whose aim is to combine more elementary operators.

- Combinatorial expression

Definition : The combinatorial expression will be represented by X, Y, Z, U, V, T...; the variables by x, y, z, t...

(i) the atomes is the combinatorial expressions

(ii) If X and Y are the combinatory expression, then (XY) is a combinatory expression.

<u>Comment</u>

we omit the most external parenthesis, where $XY =_{det} (XY)$.

Associativity

$XYZ =_{det} ((XY)Z) \neq X(YZ)$

Applications of CL

- In constructing the founcdations of mathematics
- In constructiuon methods and tools for implementing the programming languages => Haskell
- Working on the Combinatory Logic:

Fitch (1974)

Klop (1992, 1993)

Shaumyan (1987); Universal Applicative Grammar :application to the NLP

Desclés (1999): study of the grammatical and lexical meanings

Steedman (2000): syntax-semantic interface

Terese (2003)

Bimbó (2011)

Theory of combinators

- Combinatory base: S and K

  All combinators can be defined from the combinators S and K.

- Combinators, called elemantary : I, K, B, W, C, S, Φ, Ψ
- A combinator is a combinatorial expression which contains only the occurrences of combinators.
- Example: is combinators?

  SKK

  (S(Kx))((SK)K)

  S(KS)K

  S(SSKS)(KK)

Combinatory base: S and K

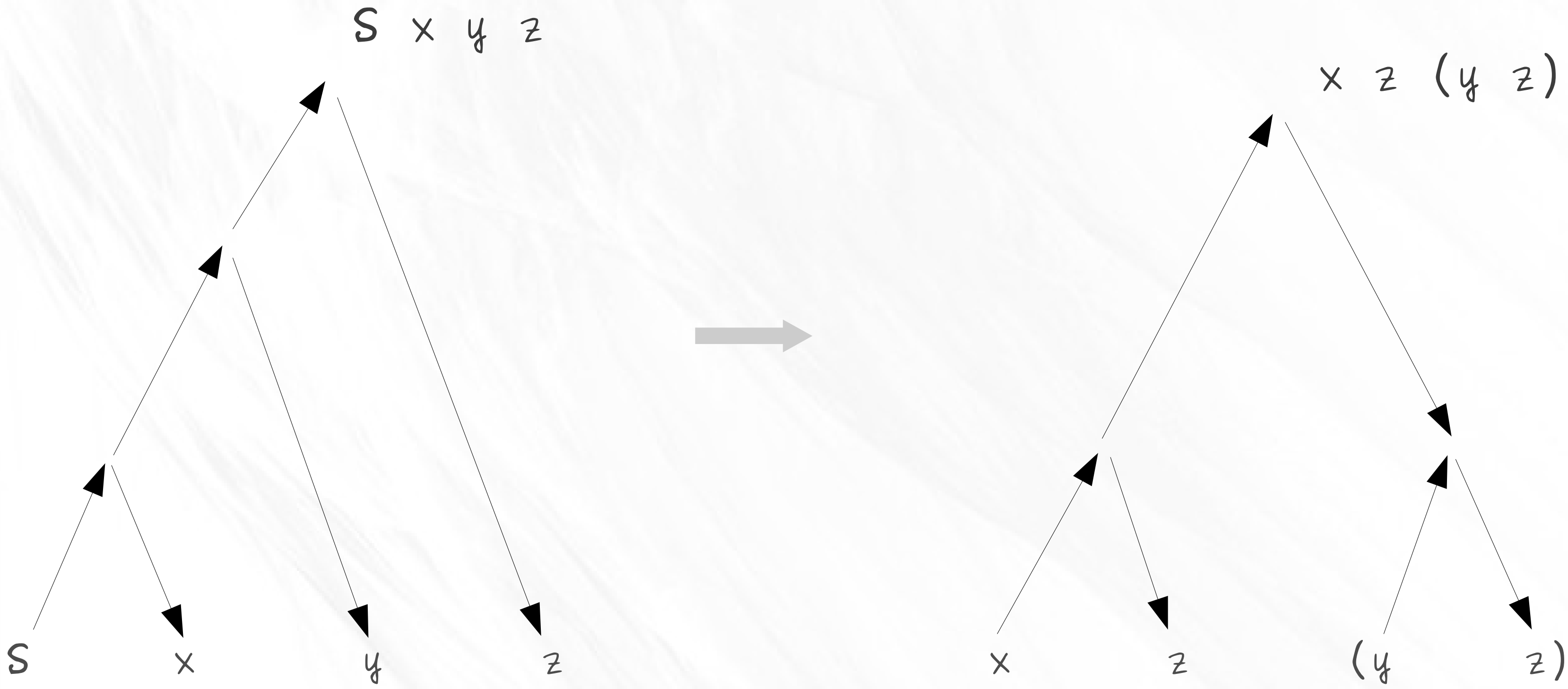
- K is defined by the rule : Kxy:=x

  the combinator K takes two arguments and returns the first argument as result. → effacement

- S is defined by the rule : Sxyz:=xz(yz)

  the combinator S composes the functions x (binary) and y (unary) with the argument z. → composition

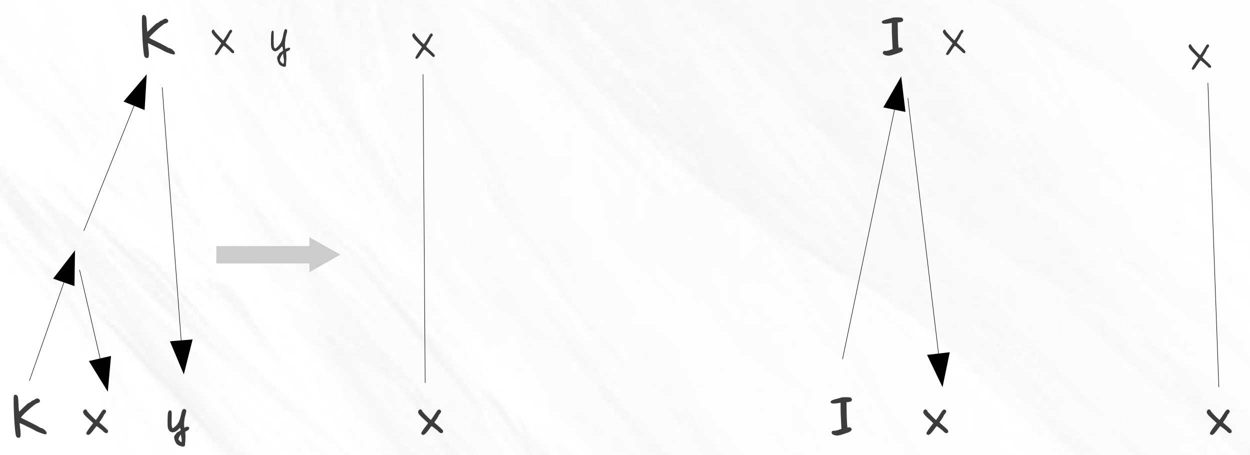
- Sxyz-> xz(yz)



- I is defined by the rule : Ix:=x

  the combiantor I takes one argument x and returns this argument as result. → identification

K x y → x   I x → x



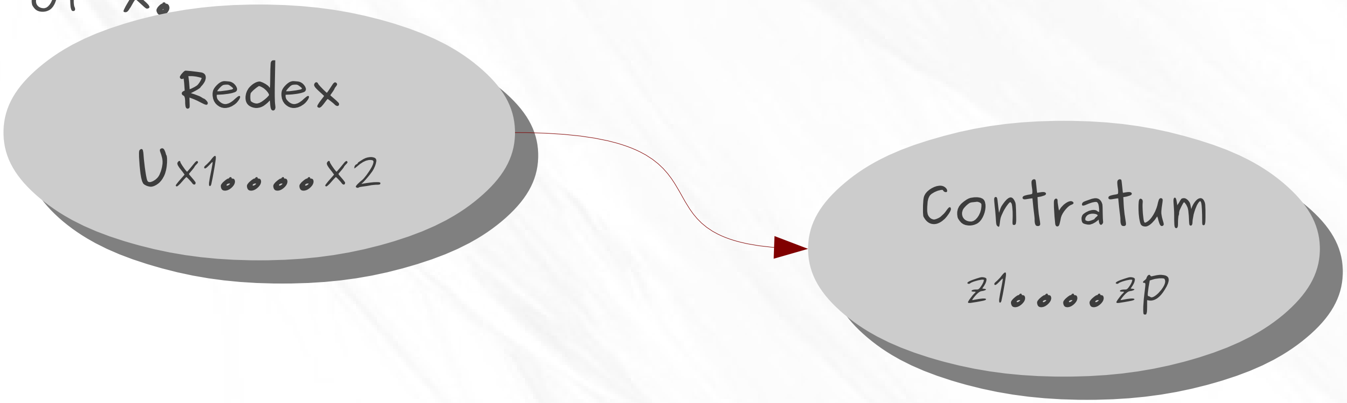
- Combinators is composable between them.
- The combinators organize an algebraic structure, for some of them, we have an algebraic tree.
- The action of combinators is intrinsic, that is, independent of the domains of the compound operators.

Normal form

- A normal form is a combinatoryal expression which can not be reduced, that is, it contains any occurences of combinators

  Definition

  If a combinatorial expression is reduced to a combinatorial expression which is in the normal form, then N is called the Normal form of X.



- Completeness of the S-K basis

S and **K** can be composed to produce combinators that are extensionnally equal to any **lambda term**, and therefore, to any computable function by Church's thesis. The proof is to present a transformation, T[ ], which converts an arbitrary lambda term into an equivalent combinator. → operation of abstraction

See the 2nd question of the classwork N°2.

Abstraction and substitution

- Two operations which construct combinatorial expressions from the combinatorial expression already defined.

(1) operation of abstraction

The expression [λx].e is a combinatorial expression which is a result of a calculus defined by the following conditions:

a. [λx].e=**K**e (condition: e does not appear in x)

b. [λx].e=**I**

c. [λx].ex=e

d. [λx].e1e2= **S**([λx].e1)([λx].e2)

(1.1) Abstraction algorithm

- An algorithm of abstraction aims to carry out the actual calculus, by abstraction of the variable x, of the combinatorial expression $[\lambda x].e$.
- Abstraction algorithms are generally presented in the form of algorithms of Markov (string rewriting system). The reasonning of the algorithm is gouverned by the following 4 metarules:

i) we apply obligatorily one rule if possible, if not we pass to the next step;

ii) we start always by trying the first step;

iii) since one rule was applied, we return to the first step;

iv) the result is obtained when any rule can be applied.

→ The algorithm of Markov given by the set totally ordered by the rules (a), (b), (c) and (d) is an algorithm of abstraction. These rules function on the combinatorial expressions.

- Example

$[\lambda x].xy$	$= S([\lambda x].x)([\lambda x].y)$	rule (d)
	$= SI\ ([\lambda x].y)$	rule (b)
	$= SI\ (Ky)$	rule (a)
	$= S\ (Ky)$	elimin. of I

(2) operation of substitution

$\lambda x.(e1\ e2)$

a function which takes an argument, say a, and substitutes it into the lambda term (e1 e2) in place of x, yielding (e1 e2)[x := a].

$(e1\ e2)[x:=a] = (e1[x:=a]\ e2[x:=a])$

$$(\lambda x.(e1\ e2)\ a) = ((\lambda x.e1\ a)(\lambda x.e2\ a))$$

$$= (S\ \lambda x.e1\ \lambda x.e2\ a)$$

$$= ((S\ \lambda x.e1\ \lambda x.e2)\ a)$$

By extensional equality,

$\lambda x.(e1\ e2) = (S\ \lambda x.e1\ \lambda x.e2)$

- Example : $[\lambda xy].x = [\lambda x.x]\ [\lambda y.x]$

$= I\ (Kx)$ rule (a and b)

$= Kx$ elimin. of I

Summing up

- The CL is a logic of the operating process by means of intrinsic compositions of operators.
- The composition is intrinsic when it is independent from domains of the operators (thus of their extensionnelles meanings).
- The CL allows to build operators and complex predicates from operators and from more elementary predicates.
- The combinators of the CL is operators of " intrinsic composition ".

Next week...

- More about the combinators: elementary and complex.