CS 524 Lecture 7

José Meseguer

The Logic of Parallel Functional Programming

(for Combinatory Logic)

Since combinatory logic (CL) is a machine-friendly formalism to express functional programs, it can be implemented in the usual way in a sequential implementation. Of course, we can then implement a more user-friendly language by first translating it into CL [or, more generally, in an extension of CL by other more powerful combinators, called super-combinators (see the book of Peyton Jones)] and then using the compiler for CL. From this point of view, CL and its extensions is a very useful intermediate language. For example OCaml is an extension of Caml, which was originally an implementation of the ML language by translation into categorical combinators, which are combinators associated to the algebraic axiomatization of cartesian-closed categories,

which are themselves categorical models of the [typed] Lambda Calculus.

However, CL itself [and as we shall see later also the Lambda Calculus] is an intrinsically concurrent formalism. I will use the word "parallel" instead of "concurrent" because, although the meaning is the same, "parallel" is used more often for deterministic [single-valued] as opposed to non-deterministic computations. So, as with all other examples, the key question is:

What is the logic $\mathcal{L}(CL)$ of parallel functional computations for CL?

Let us, by abuse of language, denote by CL not only it syntax and inference rules [specifiable in Maude] but also the set of its expressions, so we can write $t, t', u, v, w \in CL$ ranging over such expressions. Then, $\mathcal{L}(CL)$ is defined as the logic where

Parallel functional CL computation = Deduction in $\mathcal{L}(CL)$

As follows:

Idle/Reflexivity

$$\frac{}{t \xrightarrow{t} t} \text{ if } t \in CL$$

Congruence

$$\frac{t \xrightarrow{\alpha} t' \quad u \xrightarrow{\beta} u'}{t\,u \xrightarrow{\alpha\beta} t'\,u'}$$

Replacement

$$\frac{t \xrightarrow{\alpha} t' \quad u \xrightarrow{\beta} u'}{(K\,t)\,u \xrightarrow{\text{K-red}(\alpha,\beta)} t'}$$

$$\frac{t \xrightarrow{\alpha} t' \quad u \xrightarrow{\beta} u' \quad v \xrightarrow{\gamma} v'}{((S\,t)\,u)\,v \xrightarrow{\text{S-red}(\alpha,\beta,\gamma)} (t'\,v')\,(u'\,v')}$$

$$\frac{t \xrightarrow{\alpha} t'}{I\,t \xrightarrow{\text{I-red}(\alpha)} t'}$$

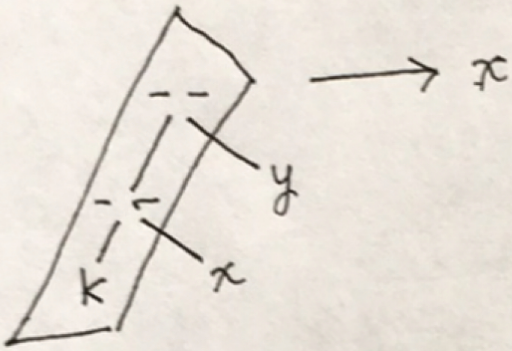
Transitivity

$$\frac{t \xrightarrow{\alpha} t' \quad t' \xrightarrow{\beta} t''}{t \xrightarrow{\alpha;\beta} t''}$$

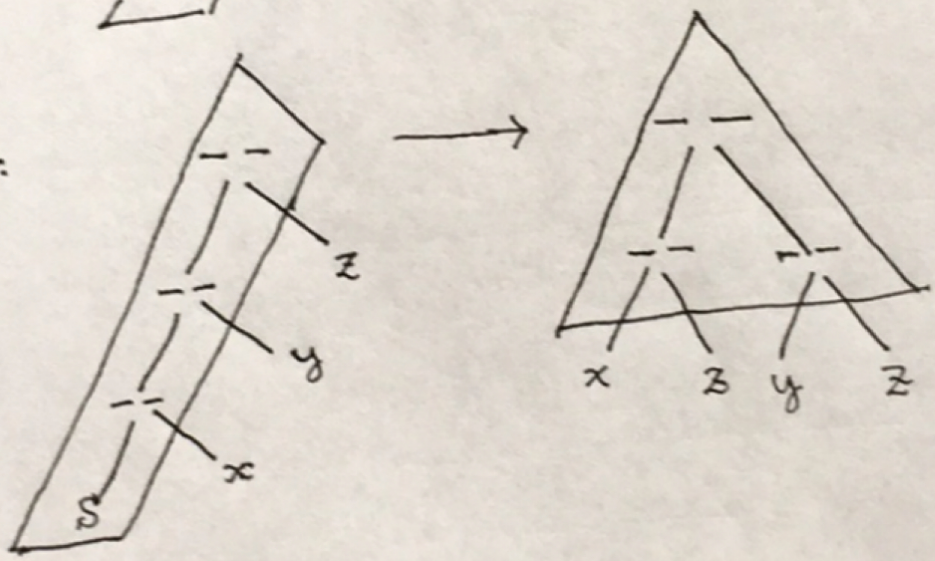
In which sense does $\mathcal{L}(CL)$ model parallel functional computation?

To gain a better intuitive understanding it can be helpful to view CL expressions as trees, and therefore the CL rules as tree transformation [rewriting] rules:

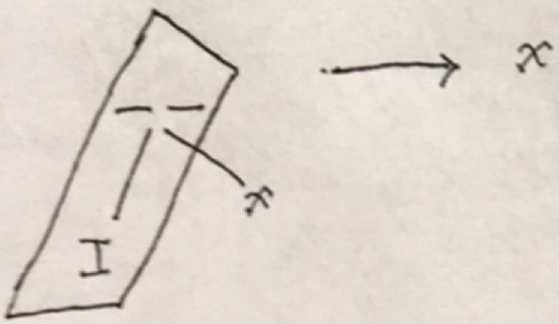
K-red:



S-red:



I-red:



Then, for example, for any $u, v \in CL$, the proof tree:

$$\dfrac{\dfrac{\dfrac{}{I}}{u \xrightarrow{u} u} \quad \dfrac{\dfrac{}{I}}{v \xrightarrow{v} v}}{R\ \dfrac{}{(Ku)v \xrightarrow{K\text{-red}(u,v)} u}} \qquad \dfrac{\dfrac{\dfrac{}{I}}{u \xrightarrow{u} u}}{R\ \dfrac{}{Iu \xrightarrow{I\text{-red}(u)} u}} \qquad \dfrac{\dfrac{\dfrac{}{I}}{v \xrightarrow{v} v}}{R\ \dfrac{}{Iv \xrightarrow{I\text{-red}(v)} v}}$$

$$\dfrac{}{R\ \dfrac{}{(((S((Ku)v))(Iu))(Iv) \xrightarrow{S\text{-red}(K\text{-red}(u,v),I\text{-red}(u),I\text{-red}(v))} (uv)(uv)}}$$

corresponds to [models] the parallel functional computation: