㉓ Defining Concurrent Processes Constructively

Y.Takayama(沖電気工業，日本)

発表要旨

Concurrent Functional Programmingとは，構成的論理の一種でλ−Calculusにstreamを加えた体系である。

構成的プログラミングとは，仕様として書いた論理式の証明図を作成し，そこからプログラムを抽出する枠組みであるが，streamを扱うためにはstreamおよびその型を論理的枠組とプログラムの中でどう定義するかを考えなければならない。またstreamに関する推論規則も必要である。

そのために，本稿ではstreamの概念をプログラムレベルと証明レベルとに分けて定義した。プログラムレベルでは，streamを無限のリストとして捕えるが，証明レベルでは，streamをBrowerのchoice sequence,その型をNat→σとして捕えた。streamをquantifyした式の意味はOpen Data PrincipleおよびFunction Continuity Principleによって与えられると考え，streamに関する推論規則としてMPST規則を導入した。以上のことから，構成的プログラミングの枠組で，streamを扱うプログラムが表現できるようになった。

質疑応答

質問：本研究で与えられた理論と，従来のclassical recursion theoryとの関係を，infinite computabilityという点から説明してください。

回答：本研究では，プログラムの性質を記述するのが主な目的であるので，可能な限りプログラムの実行時の振舞いに関することは排除するようにしています。プログラムレベルで，recursive function theoryと本研究とは密接な関係がありますが，しかしそれはlogical partのそれとは独立です。ですから，たぶんrecursive function theoryは，論理の部分でも関係があると思われますが，それは概念的には全く独立のものです。

質問：extensionのところで，Nondeterminacyについて取り上げていましたが，この体系でstreamをmargeするようなものは書けるでしょうか。

回答：その例題については，Proceedingsにも書きましたが，今説明した規則を使うと簡単に書くことができますが，これには少し問題があります。即ち，推論規則の制限が強いので，この規則ではnondeterminacy margerの仕様を詳しく書くことは不可能で，それより少し弱い仕様だけが記述できるようになっています。しかし，仕様に適切な証明を持ってくることにより，そこからstream margerを得ることができます。