A. Nerode M. Taitslin (Eds.)

Logical Foundations of Computer Science – Tver '92

Second International Symposium
Tver, Russia, July 20-24, 1992
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Contents

Modal Linear Logic

Dimitry A. Archangelsky and Mikhail A. Taitslin

62 Mojayskogo Str., Apt. 265
Tver, 170043
Russia

Abstract

In this paper we continue our study of Girard's Linear Logic and introduce a new Linear Logic with modalities. Our logic describes *not only consumption*, *but presence* of resources as well. It describes transformation of resources not only for the single point but for some net, where supplies can be sent from one object to another one using interfaces. We introduce a new semantics and a new calculus for this logic and prove the completeness theorem for this calculus in respect to our semantics.

1 Introduction

This paper continues the study of Girard's Linear Logic, began by Girard in [1,2] and developed by Abramsky [3], Lafont [4],[5], Lincoln, Mitchell, Scedrov, and Shankar [6], and Kanovich.

In Linear Logic the statement "Γ implies Δ" means that *presented resources* Γ can be transformed into resources Δ, being spent completely. Although programming interpretations of Linear Logic were devoted to describing this situation, in our view these attempts did not achieved their declared aim. In previous formalizations the condition of resource presence is not, in fact, taken into account. In our formalization this gap is eliminated. Note that, in correspondence with the traditional Linear Logic approach, the consumption of converters is also taken into account.

Also our formalization describes each concrete situtation, but 'not' properties are true always.

The real situation is not that an object acts separately, but that a net of connected objects circulates supplies. Usually acceptable tools of transformation of supplies are in fixed places and the net does not transmit them. Our formalization is an attempt to reflect this.

We introduce some restrictions on the architecture of the nets, supposing that the net has the tree form. This corresponds to the modern point of view on computer net organization. Computers used as terminals may use their own supplies as well as the

supplies of some larger computer with which they may be directly connected. In turn this larger computer and others similar to this one may be connected with some bigger computer and so on. Supplies may be transmited not only from some directly connected computer, but also from a more remote computer via a link of direct connections.

All previous definitions of Linear Logic were based on constructing systems of axioms and did not propose any theoretical-model semantics. We have attempted to eliminate this omission as well.

2 The model

We fix a finite set S. The elements of S will be called supplies. We fix a finite set Pr. The elements of Pr have the form $X \to Y$, where X and Y are sequences of supplies. The elements of C will be called basic converters. The elements of the union $S \cup Pr$ of the sets S and Pr are called basic resources and are denoted by R. We denote the set of all natural numbers by ω.

The model is a finite tree whose nodes are objects. The object is a mapping from R to ω. For r from R and for an object α, $\alpha(r)$ denotes the number of copies of the resource r at a point α.

If α and β are objects and β is a direct successor of α, then we write $\beta << \alpha$. Let $\beta << {}^1\alpha$, if $\beta << \alpha$ or β is α, and let, for $i > 0, \beta << {}^{i+1}\alpha$ be $\beta << {}^1\gamma$ and $\gamma << {}^i\alpha$ for some object γ. Let $\beta << {}^0\alpha$ if β is α.

3 The language

The expressions of the language are built from symbols of the following sets:

S - the set of propositional variables called supplies;

$\vdash$ - symbol of sequent;

$\to$ - symbol of the binary propositional linear implication operation;

$\square$ - symbol of modal operation;

(,) - brackets.

The set *SList* of supply lists is the least satisfying the following conditions:

$S \subseteq SList$;

if $A, B \in SList$, then $(AB) \in SList$.

The set *For* of formulas is the least satisfying the following conditions:

$S \subseteq For$;

if $A, B \in For$ then $(AB), (A \to B), \square A \in For$.