Petri Nets, Horn Programs, Linear Logic, and Vector Games

Max I. Kanovich

Russian Humanities State University, Moscow, Russia
email: sergei@artemov.mian.su

Abstract. Linear Logic was introduced by Girard as a resource-sensitive refinement of classical logic. Linear Logic is of considerable interest for Computer Science. In this paper we focus on the correlations between natural fragments of Linear Logic and a number of basic concepts related to different branches of Computer Science such as Concurrency Theory, Theory of Computations, Horn Programming, and Game Theory. In particular, such a complete correspondence allows us to introduce several new semantics for Linear Logic and to clarify many results on the complexity of natural fragments of Linear Logic.

1 Introduction and Summary

Linear Logic was introduced by J.-Y. Girard [12] as a resource-sensitive refinement of classical logic. From a purely logical point of view, Linear Logic can be conceived of as a substructural logic in Gentzen-style sequential formalisms: Structural rules such as Contraction and Weakening (which are not sensitive to aspects of control) are deleted. Many of intricate applications of Linear Logic in Computer Science owe their existence to the lack of these structural rules. In particular, an exact correspondence has been established between natural fragments of Linear Logic and natural complexity classes [22, 18]. In this paper we focus on the study of a number of basic concepts related to different branches of Computer Science: Concurrency Theory, Theory of Computations, Horn Programming, and Game Theory, tied together with some natural fragments of Linear Logic.

From a logical point of view, we will study the derivability problem in Linear Logic for sequents of the Horn structure, the so-called (!,|)-Horn sequents, i.e. sequents of the form\(^1\)

\[ W, \Gamma \vdash Z, \]

where \( W \) and \( Z \) are tensor products of positive literals,\(^2\) \( \Gamma \) is a multiset consisting of Horn implications

\[ (X \rightarrow Y), \]

\(^1\) Where \( !\Gamma \) stands for the multiset resulting from putting the modal storage operator \(!\) before each formula in \( \Gamma \).
\(^2\) Henceforth, such tensor products will be called simple products.
and $\oplus$-Horn implications

$$(X \rightarrow (Y_1 \oplus Y_2 \oplus \cdots \oplus Y_m)),$$

here (and henceforth) $X, Y, Y_1, Y_2, \ldots, Y_m$ are simple products.

From a programming point of view, we study Horn programs, taking into account resources consumed in computations.

Definition 1. A Horn program is an acyclic directed finite graph each edge of which is labelled by a Horn implication of the form $(X \rightarrow Y)$. Such an implication describes the elementary assignment operation of producing $Y$ from $X$ which is performed on the corresponding edge. Vertices with no incoming edges are specified as input ones. Vertices with no outgoing edges are specified as output ones.

In order to control the use of resources, we interpret each of our Horn implications $(X \rightarrow Y)$ as the following assertion:

- Given $X$, $Y$ can be computed,
- and $X$ is consumed in the process of this computation,
- so that the old values of $X$ are no more available.

Given an input $W$, an output $Z$, and a set of instructions $\Gamma$, our problem is to construct a Horn program for producing $Z$ from $W$. Each elementary operation of this program should consist in implementing a certain Horn implication taken from $\Gamma$.

From a concurrency point of view, we study the reachability problem in generalized Petri nets. Such a Petri net can include:

(i) Ordinary transitions. The firing of such a transition yields deterministic rearrangement of the current marking of the Petri net.

(ii) Non-deterministic transitions. The firing of a non-deterministic transition involves development of our concurrent process in one of several alternative directions.

The difference between ordinary and non-deterministic transitions can be perceived as follows: When we fire a certain transition, we make thereby our own choice from the given set of possible transitions.

(i) If we choose an ordinary transition to be fired, we know the result of this firing to be deterministic.

(ii) On the contrary, having chosen a non-deterministic transition to be fired, we meet with the non-deterministic situation: We do not know in advance what direction from the set of the alternative ones will be chosen at a given occasion.

This new kind of transitions is introduced in this paper.