MAKING NETS ABSTRACT AND STRUCTURED

Ludwik Czaja
Institute of Informatics, University of Warsaw
PKiN VIIIp. 00-901 Warsaw, Poland

Abstract

Nets considered here are "abstract" since their places are seen as individual variables, transitions as function names and a firing rule as a predicate stating when a partial function, denoted by a given transition, is defined. Three selected rules expressing the local character of transition's activity are admitted as axioms for interpretation. The nets are "structured" because they can be built up of simpler nets by a concurrency constructor (||) with handshaking. From axioms we infer a necessary and sufficient condition for a net to be decomposable with respect to a given partition of places. In particular, we examine decomposition into atomic, not further decomposable nets. A number of examples illustrate the considerations.

1. Introduction

By abstract net is meant here a structure similar to the Petri net with undirected arcs connecting places and transitions alternately. The interpretation, or "firing rule", however, is quite general: arbitrary objects may be assigned to places and arbitrary transformations on "markings" - to transitions. Certainly, various types of Petri nets may be expressed in this setting by specifying a particular interpretation. But also structures such as arithmetic or boolean expressions, sequential flowchart schemata, data-flow systems etc. can be represented as abstract nets. The representation however involves graphical representation - amorphic collections of lines usually hardly intelligible, perhaps apart from simple structures. But there is a way to build up large nets from simple, easy to understand parts, by making use of suitably chosen operators on nets. We choose here a synchronised concurrency operator || of the meaning analogous to that from CSP [Boo 81]: a net P||Q, introduced in Section 2, is obtained from nets P and Q by taking them as one in which every two
transitions common to P and Q are "glued" together. In Section 3 there is a simple example of net construction by means of \( I \)-operator. Our main concern is decomposition of nets with respect to \( I \)-operator and this is the subject of Section 4. But \( I \)-decomposability is the issue of mutual dependency of places (determined by a particular interpretation) rather than of information flow and this is why we take undirected lines as arcs. Although \( I \)-decomposition has been treated extensively in the literature (see e.g. [Bra 80] and recently [Maz 84]), it concerned nets with "Petri-like" interpretations, in which case the question of \( I \)-decomposability has a rather trivial answer (Example 4.2). It turns out that such nets are \( I \)-decomposable in any possible way, but usually this is not so with other interpretations. Thus, apparent similarity of "abstract" nets to high-level Petri nets, where places hold individual, also "abstract" tokens (e.g. [Gen-Laut 81], [Jen 81], [Rei 85]) is misleading - at least as far as \( I \)-decomposability is concerned. The inability, in general, of nets with ordinary "Petri-like" interpretations, to compute even simple functions (compare Examples 4.2 and 4.3), motivated our concern with "abstract" interpretation. This, in turn, set up the question of \( I \)-decomposability. The main result is Theorem 4.3: it establishes a necessary and sufficient condition for an abstract net to be \( I \)-decomposable wrt a given partition of places. Theorem 4.4 states uniqueness of an ultimate \( I \)-decomposition, the decomposition into atomic, not decomposable subnets.

2. Abstract nets and their parallel combination

2.1 Abstract nets

An abstract net is a system \( P = \langle S,T,F \rangle,A,I \rangle \) where \( \langle S,T,F \rangle \) is a net-schema in which \( S \) is a non-empty set of variables, \( T \) is a set of operators and \( FC\{ (s,t): s \in S, t \in T \} \) is a bipartite relation called here a locality relation. An interpretation of the schema is determined by a set \( A \) and a mapping \( I \) as follows. \( A \) is a set of values of variables, a total function \( M: S \to A \) is a valuation of variables, \( N=A \) is the set of all valuations, \( I \) is a mapping, which with every operator \( t \in T \) associates a binary relation in \( N \), i.e. \( I(t) \subseteq N \times N \). \( I(t) \) will be written \( t \) and called interpretation of \( t \). There are some restrictions imposed on interpretation and some conventions. Firstly, we assume here \( t \) to be a partial function \( \widetilde{I}: M \to M \) and thus write \( M'=t(M) \) whenever \( (M,M') \in t \). If \( t \) is undefined for \( M \in M \), i.e. if there is no \( M' \) such that \( (M,M') \in t \), we write \( t(M)=\bot \) and assume \( M' \).