Chapter 7

OBJSA NETS:
OBJ AND PETRI NETS FOR SPECIFYING CONCURRENT SYSTEMS

E. Battiston

F. De Cindio

G. Mauri

Abstract
This paper provides an intuitive presentation of OBJSA nets, a specification language which combines a specific class of Petri nets, namely Superposed Automata nets, and the well-known algebraic specification language OBJ. The presentation is particularly addressed to people confident with algebraic specification techniques and focuses on the composition mechanism which allows the designer to obtain the OBJSA specification of a system by combining the specifications of the system components.

1 COMBINING PROCESS ABSTRACTION AND DATA ABSTRACTION FOR REAL SYSTEM SPECIFICATION

When using formalisms based on abstract data type techniques [14, 16, 23, 32] for real system specification, a major limitation turns out to be the lack of features for treating concurrency, synchronization, mutual exclusion and other analogous situations. These points of weakness are outlined both in the literature (see, e.g. [1]) and in concrete experiences of use of algebraic languages.
On the other hand, the most popular formalisms which allow an appropriate handling of concurrency and support process abstraction, such as (high-level) Petri nets [15, 19], Milner's CCS [27] or specialized specification languages such as SDL for telecommunication applications, are weak in supporting data abstraction.

The increasing awareness of the limitations of both families of formalisms has given rise in recent years to several attempts of defining new theories, formalisms and languages, integrating both data abstraction and process abstraction for concurrency handling. [1] and [20] survey the most of them.

In order to overcome both limitations, in [4] we defined, in both graphical and matricial form, a new class of high-level nets, called OBJSA net systems, or OBJSA nets for short. OBJSA nets are couples consisting of a Superposed Automata (SA) net [10] and of an algebraic specification, that we give using OBJ3 [17], plus the inscription function which associates the net elements with the corresponding algebraic entities. The attention is focused, on the one hand, as in CSP [18], CCS [27], and COSY [22], on the possibility of building the net system model through composition of its (sequential non-deterministic) components: this is the reason of the choice of SA nets among the various classes of Petri nets [6]. On the other hand, we stress the use of algebraic specification techniques for describing the individual tokens flowing in the net: here the choice is in favour of a well defined and supported specification language such as OBJ3.

The resulting specification language takes full advantage of the best features of Petri Nets and algebraic specification techniques. On the one hand the integration of the algebraic specification formalism into the net model allows the definition of a new class of high-level nets, equipped with a formalism for modelling the data handled by the concurrent processes which is more powerful than in the classical classes of high-level nets. More precisely, in regard to OBJSA nets, we feel very satisfied with our choice of adopting one well developed algebraic specification language such as OBJ3 instead of redefining on nets all the algebraic notations. The major advantages consist in genericity, compositionality, reusability, executability and support tools, as discussed below.

On the other hand, the net formalism gives to the algebraic specifications a way for modelling concurrency and synchronization constraints.

This mutual advantage takes its full form when combined with the modularization and compositionality features, which are recognized as a major need in the specification of concurrent systems [7, 13].

Furthermore, in order to reduce the difficulty that system designers operating in industrial framework usually find in the autonomous development of formal specifications, the definition of OBJSA nets is tightly coupled with the development of a support environment, called ONE (OBJSA Nets Environment).