We develop a framework for parallel applications, integrating and consistently supporting the coordination and the issues of correctness and efficiency. This is achieved by exploiting the DUALITY coordination model [4], which is founded on the simplicity of UNITY formalism [1] and on the modular structures provided by the principles of communication-closed layers - CCL [2]. Thus, while stressing the separation of concerns of correctness and efficiency, we provide a more structured style of parallel programming. We extend UNITY with semantic and syntactic transformations that identify the structure of a parallel program. We make a separation between the structure of the parallel program from the organization of the parallel architecture. Specifying temporal and causal ordering of events, we address both data and process parallelism. A parallel program is derived in two phases: an architecture-independent sequence of refinements coordinated by an architecture-dependent sequence of refinements. Hence, the resulting parallel program is made modular and portable across different architectures.

The framework emphasizes the duality of causality and concurrency between specifications and mappings. The specification is an architecture-independent representation of the problem and the correctness, and illustrates the parallel applications transparency. The mapping is an abstract description of how to execute reliably and efficiently a specification on a target parallel architecture. We introduce the notion of Meta-Program $MP_S$ is a causal and concurrent, dual, composition of the mapping $M_S$ and the specification $S$. Meta-Program extends UNITY’s program-schema. It allows the representation of the causal order, apart from the temporal or sequential order, using the CCL operator “•”:

$$MP_S = \text{Causal}(M_S, S) \equiv M_S \circ S.$$ Union composition, “[]”, expresses the program nondeterminism, concurrency:

$$M_S = \text{Concurrent}(M_{S1}, M_{S2}) \equiv M_{S1} [] M_{S2}.$$ A Meta-Program, is rewritten $MP_S = \text{Causal}(M_S, S) \equiv \text{Causal}(\text{Concurrent}(M_{S1}, M_{S2}), \text{Concurrent}(S_1, S_2))$. The algebraic laws of CCL [3], extended in DUALITY, allow transformations of the Meta-Program in forms that match shared-memory or distributed parallel architectures. We obtain the DUALITY model in practice:

$$MP_S = M_S \circ S \equiv (M_{S1} [] M_{S2}) \circ (S_1 [] S_2) = (M_{S1} \circ S_1) [] (M_{S2} \circ S_2) = MP_1 [] MP_2.$$ Consistent coordination implies that if the architecture independent specification of a sequential application is $S_i$, the correspondent parallel application, $MP_i$, can reuse $S_i$’s design knowledge. The correctness of the Meta-Program is deduced from the correctness of the architecture-independent specifications, preserved at each refinement step. In addition, the correctness of the complete implementation is inferred from the individually identified program modules (e.g., communication-closed layers). This allows the reuse of some design patterns and proofs of sequential algorithms for the design of parallel programs. Therefore, it reduces overall complexity and proves the effectiveness of a more abstract model of parallel applications.

References