

When using generic operations with explicit dispatch, adding a new operator requires writing a new procedure that checks whether a given argument is one of every type. The more types, the more cumbersome this becomes. When adding a new type, you must add a new procedure for that type for every existing operator, which means adding a new predicate/consequent to every generic operator procedure. This approach is not recommended because the designer of the generic operators must be knowledgeable of every possible type.

When using data-directed dispatch and adding a new operation, the only thing that needs to happen is that each type that makes use of that operation must install its specific instantiation of that operation into the dispatch table. When adding a new type, the type must include some identifier tag for each of its kind, and install procedures for all existing operations into the dispatch table.

When using message-passing, to add a new operation, you must edit the body of the constructor for each type to include a condition that checks if the operation is this new operation, and if it is, has a way to provide the required information. To add a new type, just like with the data-directed approach, nothing has to be changed in the representation of other types. All that must occur is the new type must include in its constructor a way to provide the relevant information when it is operated on through the use of an internal dispatch function.

In a system where a new type must often be added, the data-directed or message-passing styles are recommended, as no modifications of existing types is required. In a system in which new operations must often be added, all approaches require writing a new procedure for types that will be accommodating. However, the data-directed style may be the most efficient (depending on table implementation) as it doesn't require checking for every type (explicit dispatch) or every operation (message-passing).