The "undefined" construct is now parameterized by a type. A main difference between untyped SIMPLE and dynamically typed SIMPLE is that the latter assigns a type to each of its locations and that type cannot be changed during the execution of the program. This allows for a more precise and flexible evaluation of programs, as the desired evaluation strategies of the various language constructs can be captured in the type of each location.

Declarations and Initializations

Declarations in typed SIMPLE are similar to those in untyped SIMPLE, but with the addition of type annotations. These annotations provide information about the type of the variable or constant being declared, which is used to ensure that the values assigned to these variables or constants are well-typed.

Type Annotations

Typing rules in typed SIMPLE are applied during the execution of the program, in contrast to static typing, where typing rules are applied at compile-time. This dynamic typing policy allows for more flexibility in the evaluation of expressions, as the type of an expression can be inferred at run-time, based on the values of its operands.

Dynamic Checking

The dynamic typing policy in typed SIMPLE includes checks for the evaluation strategy of the program. These checks ensure that the program adheres to the intended evaluation strategy, and that the types of expressions are consistent with the desired evaluation strategies.

Abstract Syntax

The abstract syntax of typed SIMPLE is similar to that of untyped SIMPLE, with the addition of type annotations. This allows for a more precise and flexible evaluation of programs, as the desired evaluation strategies of the various language constructs can be captured in the type of each location.

Dynamic Semantics

The dynamic semantics of typed SIMPLE are very similar to those of untyped SIMPLE, but with the addition of type checks. These checks ensure that the types of expressions are consistent with the desired evaluation strategies, and that the types of variables and constants are well-typed.

Type Enforcement

Type enforcement in typed SIMPLE is performed at run-time, as the program executes. This allows for more flexibility in the evaluation of expressions, as the type of an expression can be inferred at run-time, based on the values of its operands.