0.1 System konceptów

Reprezentacja definicji szablonu w C++ to zazwyczaj drzewa wyprowadzania[1]. Używając identycznych technik kompilatora, możemy przekonwertować koncepty do takich drzew. Posiadając to, możemy zaimplementować sprawdzanie konceptów jako abstrakcyjnych drzew dopasowań[2]. Wygodnym sposobem implementowania takiego dopasowywania jest generowanie i porównywanie zestawów wymaganych funkcji i typów (zwane *zestawami ograniczeń*) z definicji szablonów i konceptów.

Definicja konceptu to zestaw równań *drzewa AST*[3] z założeniami typu. Koncepty dają dwa zamysły:

1. w *definicjach szablonu*, koncepty działają jak reguły osądzania typowania. Jeśli *drzewo AST* zależy od parametrów szablonu i nie może być rozwiązane przez otaczające środowisko typowania, wtedy musi się pojawić w strzegących ciałach konceptów. Takie zależne *drzewa AST* są domniemanymi parametrami konceptów i zostaną rozwiązane przez sprawdzanie konceptów w momentach użycia.

2. w *użyciach szablonów*, koncepty działają jak zestawy predykatów, które argumenty szablonu muszą spełniać. Sprawdzanie konceptów rozwiązuje domniemane parametry w momentach inicjalizacji.

Jeśli zestaw konceptów definicji szablonu określa zbyt mało operacji, kompilacja szablonu nie powiedzie się przez sprawdzanie konceptów. Szablon jest prawie ograniczony. Odwrotnie, jeśli zestaw konceptów definicji szablonu określa więcej operacji niż potrzeba, niektóre inne uzasadnione użycia mogą również zawieźć sprawdzanie konceptów. Szablon jest nad ograniczony. Przez "inne uzasadnione" rozumie się, że sprawdzanie typów udałoby się w przypadku braku sprawdzania konceptów.

[1] (ang. Parse Trees)
[2] (ang. Abstract Tree Matching)
[3] (ang. Abstract Syntax Tree)