Enable variable template template parameters

Document #:	D2008R0
Date:	2019-12-25
Project:	Programming Language C++ Evolution Working Group Incubator
Reply-to:	Mateusz Pusz (Epam Systems) <mateusz.pusz@gmail.com>

Contents

1 Introduction

In C++ we can easily form a template that will take a class template as its parameter. With C++14 we got variable templates. They proved to be useful in many domains but still are not first class citizens in C++. For example they cannot be passed as a template parameter to a template. This document tries to address this.

2 Motivation and Scope

To check if the current type is an instantiation of a class template we can write the following type trait:

```cpp
template<typename T>
struct is_ratio : std::false_type {};

template<intmax_t Num, intmax_t Den>
struct is_ratio<std::ratio<Num, Den>> : std::true_type {};
```

A family of such type traits can be passed to a concept:

```cpp
template<typename T, template<typename> typename Trait>
concept satisfies = Trait<T>::value;
```

and then be used to constrain a template:

```cpp
template<satisfies<is_ratio> R1, satisfies<is_ratio> R2>
using ratio_add = _see below_;
```

After C++14 we learnt that variable templates are less to type (no need for a `XXX_v` helper) and often are faster to compile. The above `is_ratio` type trait can be rewritten as:

```cpp
template<typename T>
inline constexpr bool is_ratio = false;

template<intmax_t Num, intmax_t Den>
inline constexpr bool is_ratio<ratio<Num, Den>> = true;
```

However, the above cannot be passed to a class template anymore. The syntax:

```cpp
template<typename T, template<typename> bool Trait>
concept satisfies = Trait<T>;
```

is not a valid C++ as of today.

3 Acknowledgements

Special thanks and recognition goes to Epam Systems for supporting my membership in the ISO C++ Committee and the production of this proposal.