Summary

This document describes how to add the fused transform algorithms proposed by N4167 and various national body comments to N4105, the C++ Parallelism TS. The semantics of the algorithms described herein were taken from NVIDIA’s Thrust algorithms library, described in N3408, and are common to other existing C++ algorithms libraries.

Additions to the Table of Algorithms

Add transform_reduce, transform_exclusive_scan, and transform_inclusive_scan to Table 1 of N4105.

Additions to <experimental/numeric> Synopsis

Add the following signatures to the namespace std::experimental::parallel::v1 in S 4.4.1 [parallel.alg.numeric.synopsis] to N4105:

```cpp
template<class InputIterator, class UnaryOperation, class T, class BinaryOperation>
T transform_reduce(InputIterator first, InputIterator last,
                   UnaryOperation unary_op,
                   T init, BinaryOperation binary_op);
```

```cpp
template<class ExecutionPolicy,
         class InputIterator, class UnaryOperation, class T, class BinaryOperation>
T transform_reduce(ExecutionPolicy&& exec, InputIterator first,
                   InputIterator last,
                   UnaryOperation unary_op,
                   T init, BinaryOperation binary_op);
```
template<class InputIterator, class OutputIterator,  
class UnaryOperation, class T, class BinaryOperation>
OutputIterator
transform_exclusive_scan(InputIterator first, InputIterator last,  
OutputIterator result,  
UnaryOperation unary_op,  
T init, BinaryOperation binary_op);

template<class ExecutionPolicy,  
class InputIterator, class OutputIterator,  
class UnaryOperation, class T, class BinaryOperation>
OutputIterator
transform_exclusive_scan(ExecutionPolicy&& exec, InputIterator first, InputIterator last,  
OutputIterator result,  
UnaryOperation unary_op,  
T init, BinaryOperation binary_op);

template<class InputIterator, class OutputIterator,  
class UnaryOperation, class BinaryOperation>
OutputIterator
transform_inclusive_scan(InputIterator first, InputIterator last,  
OutputIterator result,  
UnaryOperation unary_op,  
BinaryOperation binary_op);

template<class InputIterator, class OutputIterator,  
class UnaryOperation, class BinaryOperation, class T>
OutputIterator
transform_inclusive_scan(InputIterator first, InputIterator last,  
OutputIterator result,  
UnaryOperation unary_op,  
BinaryOperation binary_op, T init);

template<class ExecutionPolicy,  
class InputIterator, class OutputIterator,  
class UnaryOperation, class BinaryOperation>
OutputIterator
transform_inclusive_scan(ExecutionPolicy&& exec, InputIterator first, InputIterator last,  
OutputIterator result,  
UnaryOperation unary_op,  
BinaryOperation binary_op);

template<class ExecutionPolicy,  
class InputIterator, class OutputIterator,  
class UnaryOperation, class BinaryOperation, class T>
OutputIterator
transform_inclusive_scan(ExecutionPolicy&& exec, InputIterator first, InputIterator last,  
OutputIterator result,  
UnaryOperation unary_op,  
BinaryOperation binary_op, T init);
Add S 4.4.5 Transform reduce [parallel.alg.transform.reduce] to N4105:

```
template<class InputIterator, class UnaryFunction, class T, class BinaryOperation>
T transform_reduce(InputIterator first, InputIterator last,
                   UnaryFunction unary_op, T init, BinaryOperation binary_op);
```

Returns:  \( \text{GENERALIZED\_SUM} \) (binary_op, init, unary_op(*first), ..., unary_op(*(first + (last - first) - 1))).

Requires: Neither unary_op nor binary_op shall invalidate subranges, or modify elements in the range \([first, last)\).

Complexity: \( O(last - first) \) applications each of unary_op and binary_op.

Notes: \text{transform\_reduce} does not apply unary_op to init.

Add S 4.4.6 Transform exclusive scan [parallel.alg.transform.exclusive.scan] to N4105:

```
template<class InputIterator, class OutputIterator, 
          class UnaryOperation, 
          class T, 
          class BinaryOperation>
OutputIterator 
transform_exclusive_scan(InputIterator first, InputIterator last, 
                          OutputIterator result, 
                          UnaryOperation unary_op, 
                          T init, BinaryOperation binary_op);
```

Effects: Assigns through each iterator \( i \) in \([result, result + (last - first))\) the value of \( \text{GENERALIZED\_NONCOMMUTATIVE\_SUM} \) (binary_op, init, unary_op(*first), ..., unary_op(*(first + (i - result) - 1))).

Returns: The end of the resulting range beginning at result.

Requires: Neither unary_op nor binary_op shall invalidate iterators or subranges, or modify elements in the ranges \([first, last)\) or \([result, result + (last - first))\).

Complexity: \( O(last - first) \) applications each of unary_op and binary_op.

Notes: The difference between \text{transform\_exclusive\_scan} and \text{transform\_inclusive\_scan} is that \text{transform\_exclusive\_scan} excludes the \( i \)th input element from the \( i \)th sum. If binary_op is not mathematically associative,
the behavior of transform_exclusive_scan may be non-deterministic. transform_exclusive_scan does not apply unary_op to init.

Add S 4.4.7 Transform inclusive scan [parallel.alg.transform.inclusive.scan] to N4105:

```
#template<class InputIterator, class OutputIterator,
#        class UnaryOperation,
#        class BinaryOperation>
#OutputIterator
#transform_inclusive_scan(InputIterator first, InputIterator last,
#                         OutputIterator result,
#                         UnaryOperation unary_op,
#                         BinaryOperation binary_op);
#
#template<class InputIterator, class OutputIterator,
#        class UnaryOperation,
#        class BinaryOperation, class T>
#OutputIterator
#transform_inclusive_scan(InputIterator first, InputIterator last,
#                         OutputIterator result,
#                         UnaryOperation unary_op,
#                         BinaryOperation binary_op, T init);
```

Effects: Assigns through each iterator \( i \) in \([\text{result}, \text{result} + (\text{last} - \text{first}))\) the value of \( \text{GENERALIZED_NONCOMMUTATIVE\_SUM}(\text{binary\_op}, \text{unary\_op}(\text{*first}), ..., \text{unary\_op}(\text{*first + (i - result)})) \) or \( \text{GENERALIZED_NONCOMMUTATIVE\_SUM}(\text{binary\_op, init, unary\_op(*first), ..., unary\_op(*first + (i - result))}) \) if init is provided.

Returns: The end of the resulting range beginning at result.

Requires: Neither unary_op nor binary_op shall invalidate iterators or sub-ranges, or modify elements in the ranges \([\text{first, last})\) or \([\text{result, result} + (\text{last} - \text{first}))\).

Complexity: \( O(\text{last} - \text{first}) \) applications each of unary_op and binary_op.

Notes: The difference between transform_exclusive_scan and transform_inclusive_scan is that transform_inclusive_scan includes the \( i \)th input element from the \( i \)th sum. If binary_op is not mathematically associative, the behavior of transform_inclusive_scan may be non-deterministic. transform_inclusive_scan does not apply unary_op to init.