



Science and  
Technology  
Facilities Council

# PSyclone Transformations for NEMO

Rupert Ford, **Andy Porter**, Sergi Siso,  
Iva Kavcic (MO), Joerg Henrichs (ABOM)

# Overview



- Single and Region transformations
- Profiling
- OpenMP/ACC
- Finding out more

# Basic Types of Transformation

Single-node transformation acts on just one PSyIR node

**Region** transformation acts on a **list** of one or more PSyIR nodes

- Must be siblings
- No need to supply a list if acting on just one node
- Eligible nodes are numbered in the text view of the PSyIR:

```
Schedule[  
  0: Loop[type='None', field_space='None', it_space='None']  
  1: Loop[type='None', field_space='None', it_space='None']  
  2: Loop[type='None', field_space='None', it_space='None']  
  3: Loop[type='None', field_space='None', it_space='None']  
  4: Loop[type='levels', field_space='None', it_space='None']  
     Literal[value:'1']  
     Reference[name:'link1']
```

# Using a Transformation

1. Import the Python class
2. Create an instance of the transformation
3. (Optionally, `validate()` the target node(s))
4. `apply()` the transformation to the target node(s)

```
from psyclone.psyir.transformations import ProfileTrans
p_trans = ProfileTrans()

for invoke in psy.invokes.invoke_list:
    sched = invoke.schedule

    # Enclose all children of the schedule within a single profile region
    sched, _ = p_trans.apply(sched.children)
    sched.view()
```

# Profiling Transformation

- Inserts 'start' and 'stop' calls around the specified region of code
- Calls in to the [PSyData API](#)
- Actual profiling library chosen at compilation time
- Only [Return](#) nodes are forbidden within a profiling region
- User must [manually add](#) a [startup](#) and a [shutdown](#) call to appropriate locations in application

```
from psyclone.psyir.transformations import ProfileTrans
p_trans = ProfileTrans()

for invoke in psy.invokes.invoke_list:

    sched = invoke.schedule

    # Enclose all children of the schedule within a single profile region
    sched, _ = p_trans.apply(sched.children)
    sched.view()
```

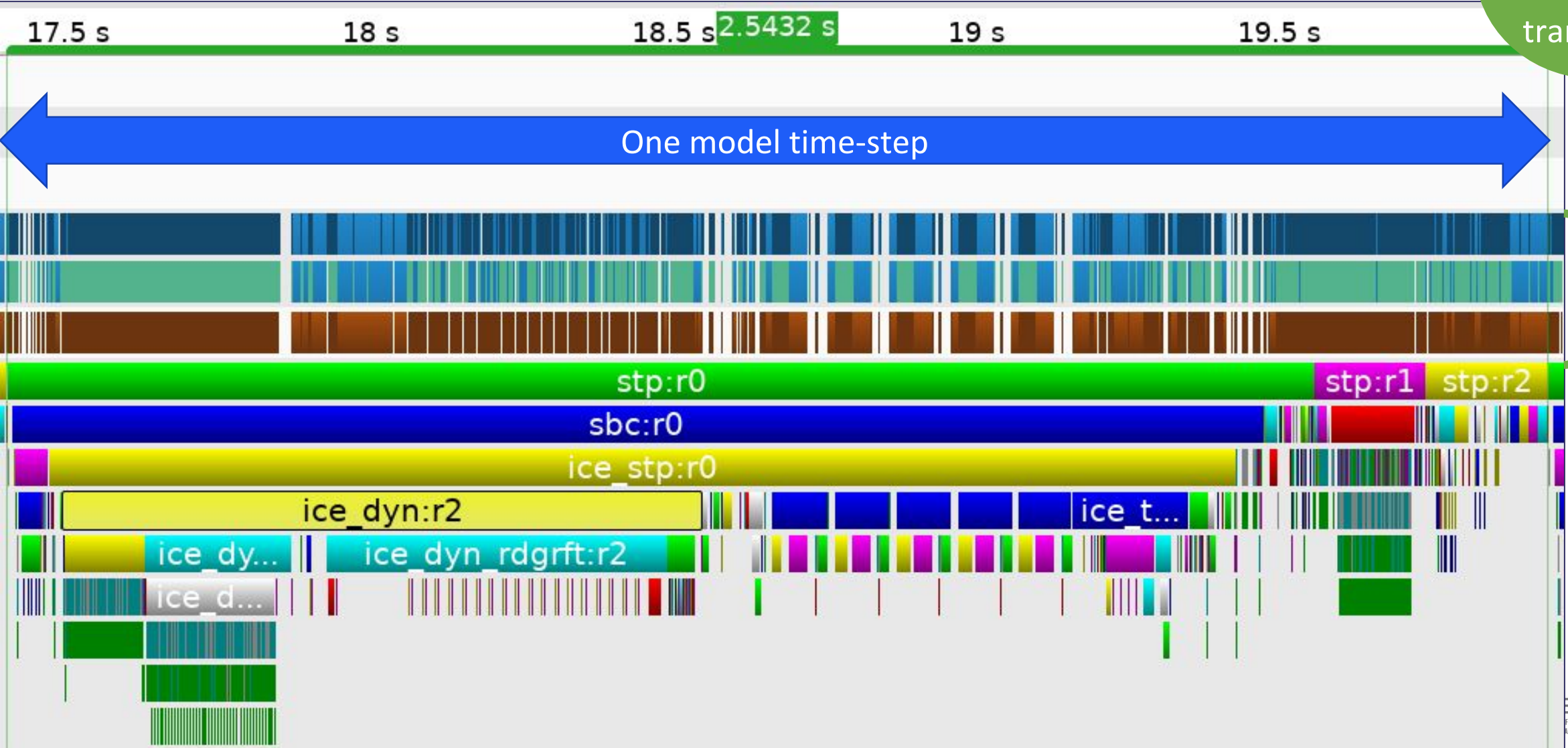
```
NemoInvokeSchedule[invoke='tra_adv_cen']
  0: Profile[]
    Schedule[]
      0: If[]
        BinaryOperation[operator='EQ']
          Reference[name='kt']
          Reference[name='kit000']
        Schedule[]
          0: If[annotations='was_single_stmt']
            Reference[name='lwp']
            Schedule[]
              0: CodeBlock[[<class 'fparser.two.Fortran2003.Write_Stmt'>]]
            1: If[annotations='was_single_stmt']
              Reference[name='lwp']
              Schedule[]
                0: CodeBlock[[<class 'fparser.two.Fortran2003.Write_Stmt'>]]
            2: If[annotations='was_single_stmt']
              Reference[name='lwp']
              Schedule[]
                0: CodeBlock[[<class 'fparser.two.Fortran2003.Write_Stmt'>]]
          1: Assignment[]
            Reference[name='l_trd']
            Literal[value='false', Scalar<BOOLEAN, UNDEFINED>]
          2: Assignment[]
            Reference[name='l_hst']
            Literal[value='false', Scalar<BOOLEAN, UNDEFINED>]
```

Can include CodeBlocks within a profiling region



# Example using NVIDIA's nvtx library

GPU  
Kernels  
and data  
transfer





# OpenMP Transformations

- OMPLoopTrans
  - Add an OMP DO directive to a loop
  - Must be within a parallel region at code-generation time
- OMPParallelLoopTrans
  - Add an OMP PARALLEL DO directive to a loop
- OMPParallelTrans
  - Create an OpenMP thread-parallel region

Transformation inserts **Directive** node

```
Literal[value:'1.', Scalar<REAL, UNDEFINED>]
17: Directive[OMP parallel]
  Schedule[]
    0: Directive[OMP do]
      Schedule[]
        0: Loop[type='levels', field_space='None', it_space='None']
          Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
          BinaryOperation[operator:'SUB']
            Reference[name:'jpk']
            Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
          Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
          Schedule[]
            0: Loop[type='lat', field_space='None', it_space='None']
              Literal[value:'2', Scalar<INTEGER, UNDEFINED>]
              BinaryOperation[operator:'SUB']
                Reference[name:'jpl']
                Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
              Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
              Schedule[]
                0: Loop[type='lon', field_space='None', it_space='None']
                  Literal[value:'2', Scalar<INTEGER, UNDEFINED>]
```

# OpenACC Transformations

ACCKernelsTrans, ACCDataTrans, ACCLoopTrans, ACCParallelTrans + others

Very similar to OpenMP apart from:

- ACCKernelsTrans

- Directs the compiler to look for parallelism and automatically offload kernels to the GPU

- ACCDataTrans

- Creates a 'structured' data region
- Used to explicitly manage the movement of data to/from the GPU memory



# Where to find out more...

ESIWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

# User Guide

psyclone.readthedocs.io

← → ↻ 🏠 <https://psyclone.readthedocs.io/en/stable/transformations.html>

PSYCLONE TRANSFORMATIONS

- PSyclone Script
- PSyIR : The PSyclone Internal Representation

☰ Transformations

- Finding
  - Standard Functionality
    - Available transformations
  - Kernels
  - Applying
- ☰ OpenMP
  - Reductions
  - Restrictions
- OpenCL
- OpenACC
- SIR

Distributed Memory

UTILITIES AND CONVENTIONS

- Stub Generation
- Line length
- Fortran Naming Conventions
- API
- PSyData API
- Profiling
- PSy Kernel Extractor (PSyKE)
- Configuration

BIBLIOGRAPHY

- Bibliography

```
class psyclone.transformations.OMPLoopTrans(omp_schedule='static') [source]
```

Adds an orphaned OpenMP directive to a loop. i.e. the directive must be inside the scope of some other OMP Parallel REGION. This condition is tested at code-generation time. The optional 'reprod' argument in the apply method decides whether standard OpenMP reduction support is to be used (which is not reproducible) or whether a manual reproducible reproduction is to be used.

Parameters: `omp_schedule` (*str*) – the OpenMP schedule to use.

For example:

```
>>> from psyclone.parse.algorithm import parse
>>> from psyclone.parse.utils import ParseError
>>> from psyclone.psyGen import PSyFactory
>>> from psyclone.errors import GenerationError
>>> api = "gocean1.0"
>>> filename = "nemolite2d_alg.f90"
>>> ast, invokeInfo = parse(filename, api=api, invoke_name="invoke")
>>> psy = PSyFactory(api).create(invokeInfo)
>>> print psy.invokes.names
>>>
>>> from psyclone.psyGen import TransInfo
>>> t = TransInfo()
>>> ltrans = t.get_trans_name('OMPLoopTrans')
>>> rtrans = t.get_trans_name('OMPParallelTrans')
>>>
>>> schedule = psy.invokes.get('invoke_0').schedule
>>> schedule.view()
>>> new_schedule = schedule
>>>
>>> # Apply the OpenMP Loop transformation to *every* loop
>>> # in the schedule
>>> for child in schedule.children:
>>>     newschedule, memento = ltrans.apply(child, reprod=True)
>>>     schedule = newschedule
>>>
>>> # Enclose all of these loops within a single OpenMP
>>> # PARALLEL region
>>> rtrans.omp_schedule("dynamic,1")
>>> newschedule, memento = rtrans.apply(schedule.children)
>>>
>>>
```

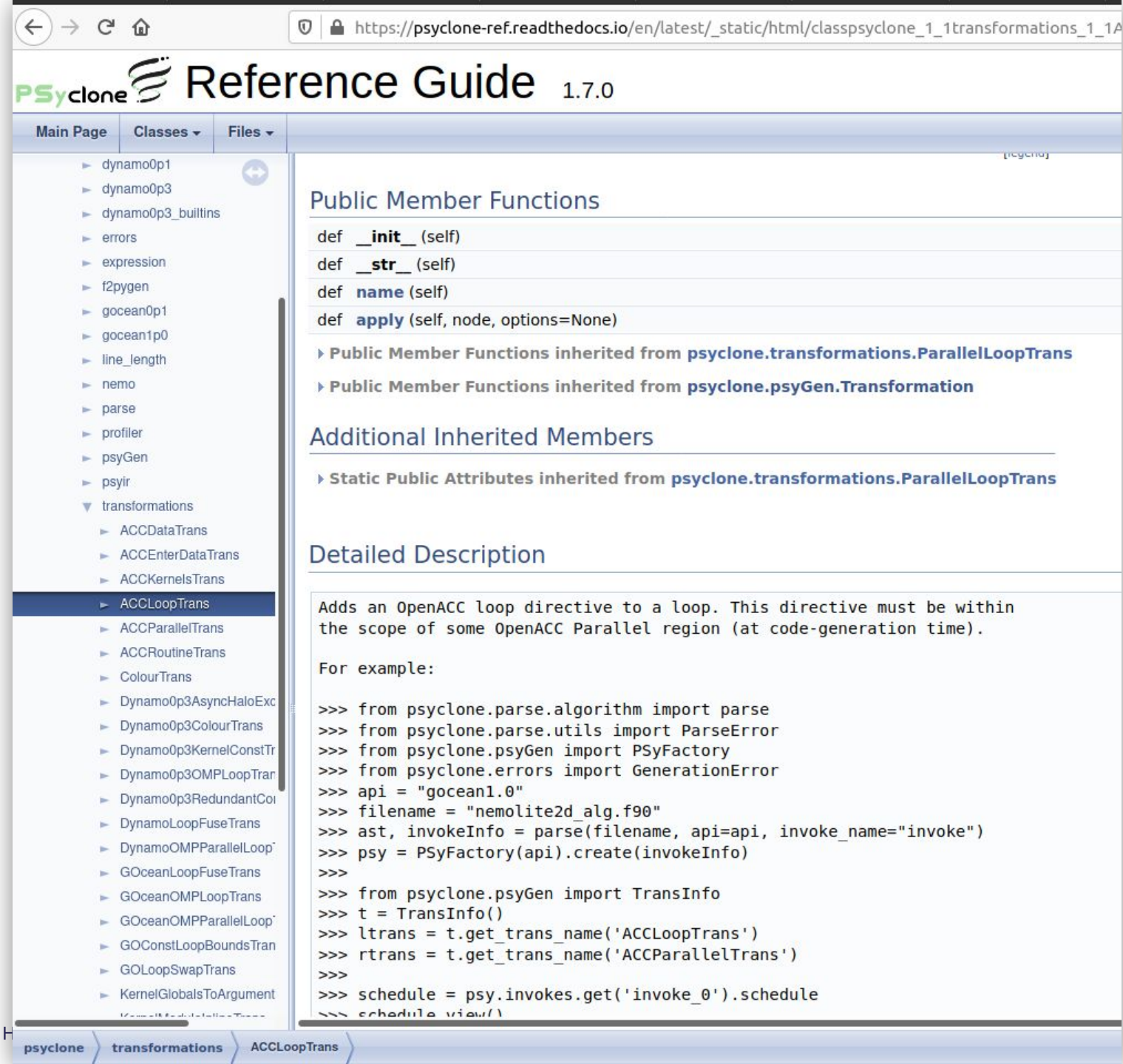
`apply(node, options=None)` [source]

Read the Docs v: stable



# Reference Guide

Doxygen-generated  
documentation available:  
[psyclone-ref.readthedocs.io](https://psyclone-ref.readthedocs.io)



The screenshot shows the PSyclone Reference Guide website for version 1.7.0. The left sidebar contains a navigation menu with categories like 'Main Page', 'Classes', and 'Files'. Under 'Classes', the 'transformations' folder is expanded, and 'ACCLoopTrans' is selected. The main content area displays the 'Public Member Functions' for 'ACCLoopTrans', including methods like `__init__`, `__str__`, `name`, and `apply`. Below this, it lists 'Additional Inherited Members' and a 'Detailed Description' section. The description states that the class adds an OpenACC loop directive to a loop and provides an example of its usage in Python code.

PSyclone Reference Guide 1.7.0

Main Page Classes Files

- dynamo0p1
- dynamo0p3
- dynamo0p3\_builtins
- errors
- expression
- f2pygen
- gocean0p1
- gocean1p0
- line\_length
- nemo
- parse
- profiler
- psyGen
- psyr
- transformations
  - ACCDATATrans
  - ACCEnterDataTrans
  - ACCKernelsTrans
  - ACCLoopTrans**
  - ACCParallelTrans
  - ACCRoutineTrans
  - ColourTrans
  - Dynamo0p3AsyncHaloExc
  - Dynamo0p3ColourTrans
  - Dynamo0p3KernelConstTr
  - Dynamo0p3OMPLoopTran
  - Dynamo0p3RedundantCoi
  - DynamoLoopFuseTrans
  - DynamoOMPParallelLoop
  - GOceanLoopFuseTrans
  - GOceanOMPLoopTrans
  - GOceanOMPParallelLoop
  - GOConstLoopBoundsTran
  - GOLoopSwapTrans
  - KernelGlobalsToArgument

### Public Member Functions

```
def __init__(self)
def __str__(self)
def name(self)
def apply(self, node, options=None)
```

Public Member Functions inherited from `psyclone.transformations.ParallelLoopTrans`

Public Member Functions inherited from `psyclone.psyGen.Transformation`

### Additional Inherited Members

Static Public Attributes inherited from `psyclone.transformations.ParallelLoopTrans`

### Detailed Description

Adds an OpenACC loop directive to a loop. This directive must be within the scope of some OpenACC Parallel region (at code-generation time).

For example:

```
>>> from psyclone.parse.algorithm import parse
>>> from psyclone.parse.utils import ParseError
>>> from psyclone.psyGen import PsyFactory
>>> from psyclone.errors import GenerationError
>>> api = "gocean1.0"
>>> filename = "nemolite2d_alg.f90"
>>> ast, invokeInfo = parse(filename, api=api, invoke_name="invoke")
>>> psy = PsyFactory(api).create(invokeInfo)
>>>
>>> from psyclone.psyGen import TransInfo
>>> t = TransInfo()
>>> ltrans = t.get_trans_name('ACCLoopTrans')
>>> rtrans = t.get_trans_name('ACCParallelTrans')
>>>
>>> schedule = psy.invokes.get('invoke_0').schedule
>>> schedule.view()
```

psyclone transformations ACCLoopTrans





Science and  
Technology  
Facilities Council

# Hands-on and questions...

```
$ git clone --recursive https://github.com/stfc/PSyclone.git  
$ cd PSyclone  
$ pip install .  
$ cd tutorials/practicals/nemo/2_nemo_profiling
```

Questions on Slack:

[https://join.slack.com/t/meteoswiss-group/shared\\_invite/zt-j7ry1st0-4TW0D9B\\_auq7tDa4zylrqQ](https://join.slack.com/t/meteoswiss-group/shared_invite/zt-j7ry1st0-4TW0D9B_auq7tDa4zylrqQ)

Andy Porter

andrew.porter@stfc.ac.uk