Introduction to DSLs

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

What is a Domain Specific Language in scientific computing ?

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Matrix transpose in different languages

matlab

C

```
for(int i=0; i < n; ++i) {
    for(int j=0; j < n; ++j) {
        double z=m[i][j];
        m(i,j)=m[j][i];
        m[j][i]=z;
    }
}
```

Matrix transpose in different languages

matlab

```
A=B.'
```

C

```
for(int i=0; i < n; ++i) {
    for(int j=0; j < n; ++j) {
        double z=m[i][j];
        m(i,j)=m[j][i];
        m[j][i]=z;
    }
}
```

Matrix transpose in different languages

matlab

A=B.'

Domain specific language:
domain = matrix operations
Semantic information: matrix transpose

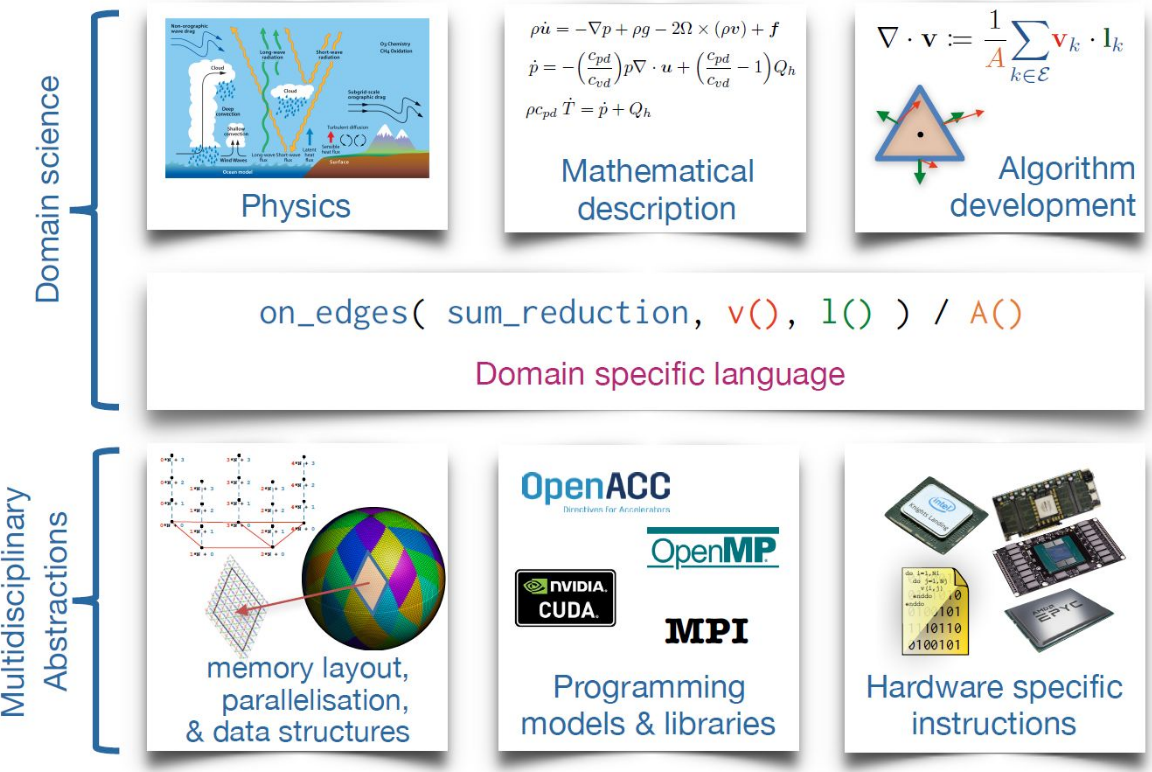
Concise syntax for a problem
Abstracts implementation and hardware dependent details:
- ✓ OMP parallelization
- ✓ GPU cuda implementation
- ✓ linear algebra calls (blas, mkl,...)

C

```
for(int i=0; i < n; ++i) {
   for(int j=0; j < n; ++j) {
      double z=m[i][j];
      m(i,j)=m[j][i];
      m[j][i]=z;
   }
}
```

General purpose, it can solve any problem
Semantic information: double nested loop modifying matrix with self-dependencies

DSLs for weather and climate models



- **Concise language** for solving weather problems
- **High scientific productivity**
- Leverage **high-level semantic** of the problem to apply **domain specific optimizations.**

W. Deconinck (ECMWF)

From math to implementation:

$$grad f = \nabla f$$

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

From math to implementation:

$$grad f = \nabla f$$

```
CALL halo_exchange(psi_c, start_prog_area, start_prog_area+1)

#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

From math to implementation:

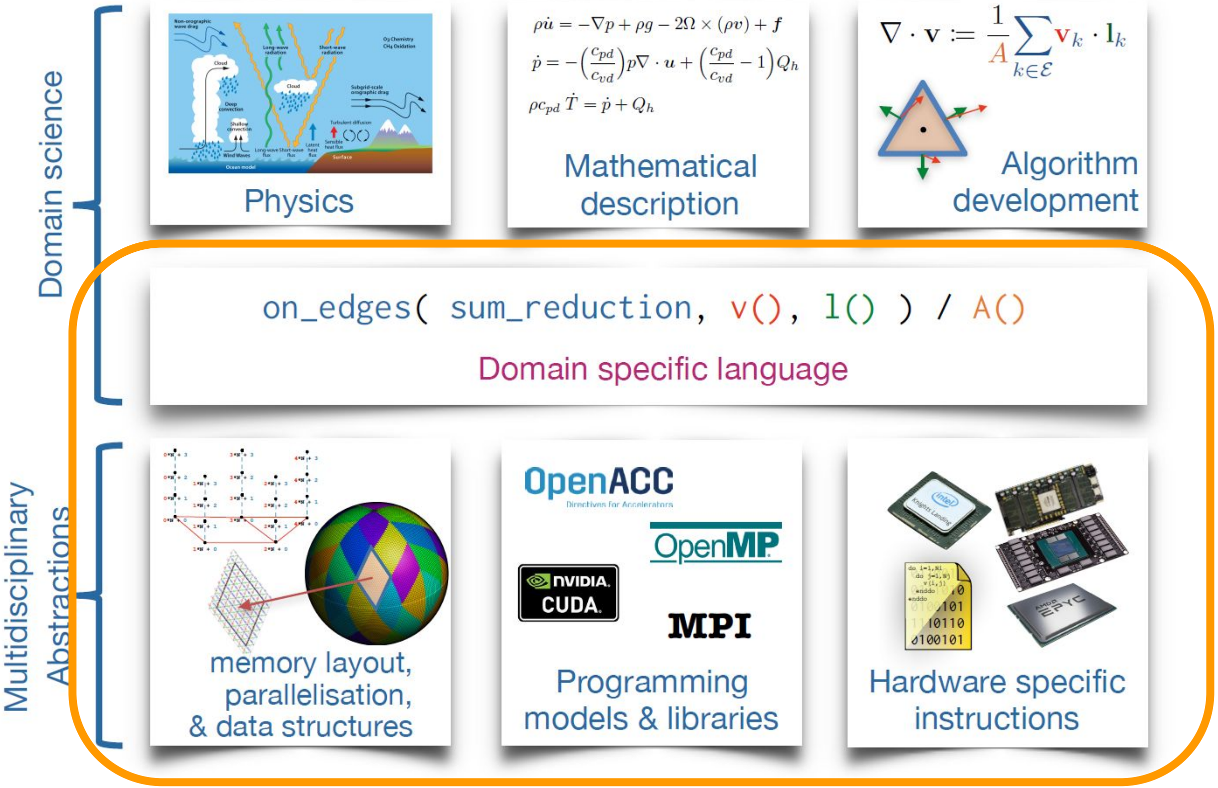
$$grad f = \nabla f$$

```
CALL halo_exchange(psi_c, start_prog_area, start_prog_area+1)

#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Separation of concerns decompose the implementation layer adding a simple interface to user (DSL syntax)



W. Deconinck (ECMWF)

Separation of concerns:

How do we separate science from complex model implementations for exascale computers ?

```
CALL halo_exchange(psi_c, start_prog_area, start_prog_area+1)

#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
       grad_norm_psi_e(je,jk,jb) =  &
         ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
           psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
         / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

Separation of concerns:

Option 1: rely on compilers

```
grad_norm_psi_e(:,:,:) =  &
        ( psi_c(iidx(2),:,:) -
          psi_c(iidx(1),:,:))
        / lhat(:,:)
```

Caveat: unrealistic option

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Separation of concerns:

Option 2: use general purpose programming models

OpenCL™

kokkos

Caveat: provides portability, but still complex and low level implementations

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Separation of concerns:

Option 3: DSLs for weather and climate

Caveat: community needs to agree on DSL languages and develop compilers

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Types of DSLs...

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

Types of DSLs

High abstractions, provide a new syntax with keyword only for supported concepts.
Eliminate all unnecessary language elements:
for/do loops, memory patterns, indices of an iteration space, parallelization

```
grad_norm_psi_e =
    reduce( psi_c,
            CELL > EDGE,
            [1/lhat, -1/lhat]
    )
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Types of DSLs

Low abstractions, support existing programming languages but limit the functionality supported.
Operates on naive implementations of general purpose languages.
Requires Implement a DSL compiler that parses, interprets computational patterns, and produces optimized code.

```
! Horizontal tracer gradient
DO jk = 1, jpkm1
   DO jj = 1, jpjm1
      DO ji = 1, jpim1   ! vector opt.
         zdit(ji,jj,jk) = ( ptb(ji+1,jj  ,jk,jn) - ptb(ji,jj,jk,jn) ) * umask(ji,jj,jk)
         zdjt(ji,jj,jk) = ( ptb(ji  ,jj+1,jk,jn) - ptb(ji,jj,jk,jn) ) * vmask(ji,jj,jk)
      END DO
   END DO
END DO
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Types of DSLs

Dusk /dawn -> Python DSL language
PSyclone -> Fortran DSL language

PSyclone -> Fortran

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

GridTools: Early examples

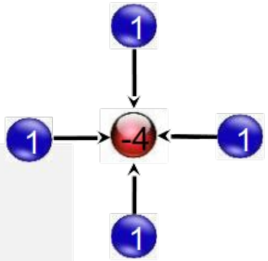
- In production at MeteoSwiss since 2016 for COSMO dynamical core.
- First operational case successfully using DSLs for dynamical core.
- Low level C++ syntax.
- Requires an HPC/C++ expert to develop model using DSL.

GridTools DSL example (in production)

- Syntax for a stencil

```
struct Laplace{
  typedef in_accessor<0, extent<-1,1,-1,1> > u;
  typedef out_accessor<1> lap;

  template <typename Evaluation>
  static void Do(Evaluation eval, domain)
  {
      eval(lap()) = eval(-4*u() + u(i+1) + u(i-1) + u(j+1) + u(j-1));
   }
};
```



- Composing stencils

```
auto hori_diff =
    make_computation<Cuda>(
        domain, grid,
        make_multistage(
            execute<forward>(),

            make_stage<Laplace>(lap(), u(), crlat0()),
            make_stage<Div>(u(), lap(), coeff())
    )
);
hori_diff->run();
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

dusk/dawn: towards new generation Python DSLs

The DSL Idea

$$\underline{\nabla}_n \psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

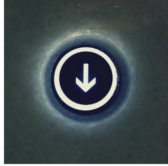
```
grad_norm_psi_e =
      reduce( psi_c,
              CELL > EDGE,
              [1/lhat, -1/lhat]
          )
```

OMP

DSL compiler

```
!$OMP PARALLEL
!$OMP DO PRIVATE(jb, i_startidx, i_endidx, je, jk)
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
!$OMP END DO NOWAIT
!$OMP END PARALLEL
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988



Existing code & DSLs

Evolution rather than Revolution

- Although DSLs are very powerful, an application must be re-written in order to use them
- Applications in the weather/climate domain are large and under continuous development
- DSLs are relatively new and untested in this domain
  - Concerns over longevity of necessary tool chains
- To stop development on existing code and re-develop from scratch is expensive (time and effort)
- Community has a lot of skill and knowledge in existing coding approaches (Fortran)

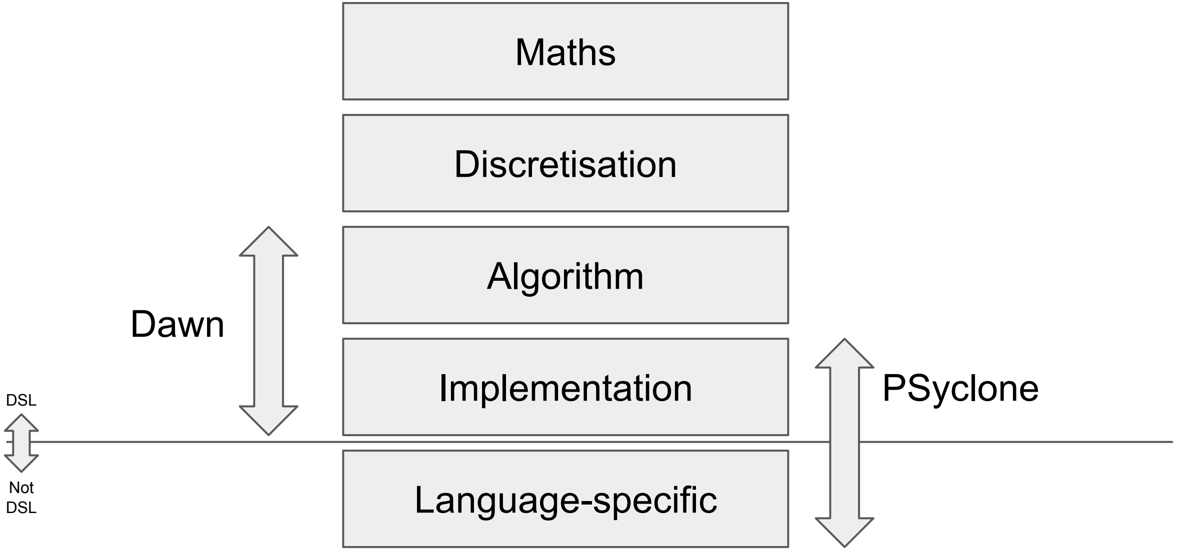
Very attractive to be able to *translate* existing code into a DSL or use existing code in a DSL rather than re-write:

- Support science that cannot be specified in the DSL language
- Transition to high level DSLs by evolution not revolution
- Support code generation and translation

Need to regain lost information

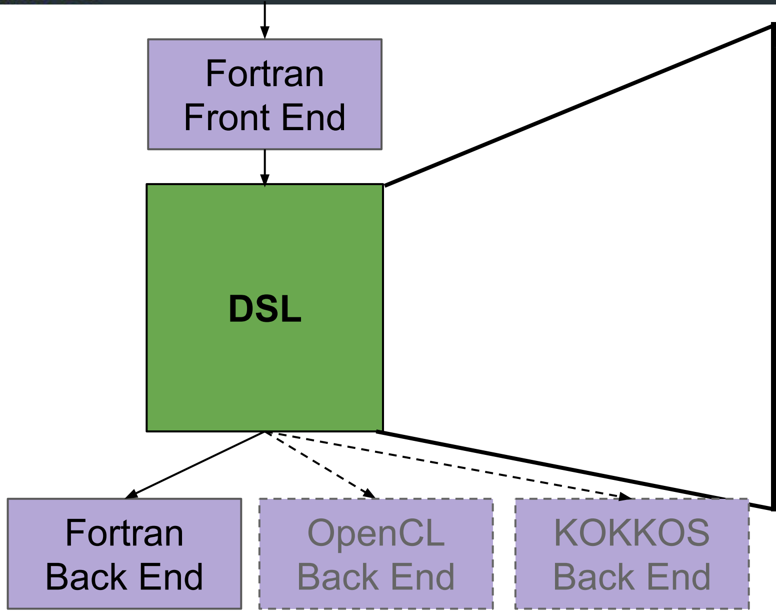
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Levels of abstraction



ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

```
! Horizontal tracer gradient
DO jk = 1, jpkm1
   DO jj = 1, jpjm1
      DO ji = 1, jpim1   ! vector opt.
         zdit(ji,jj,jk) = ( ptb(ji+1,jj  ,jk,jn) - ptb(ji,jj,jk,jn) ) * umask(ji,jj,jk)
         zdjt(ji,jj,jk) = ( ptb(ji  ,jj+1,jk,jn) - ptb(ji,jj,jk,jn) ) * vmask(ji,jj,jk)
      END DO
   END DO
END DO
```




```
4: Loop[type='levels', field_space='None', it_space='None']
    Literal[value:'1']
    Reference[name:'jpkm1']
    Literal[value:'1']
    Schedule[]
        0: Loop[type='lat', field_space='None', it_space='None']
            Literal[value:'1']
            Reference[name:'jpjm1']
            Literal[value:'1']
            Schedule[]
                0: Loop[type='lon', field_space='None', it_space='None']
                    Literal[value:'1']
                    Reference[name:'jpim1']
                    Literal[value:'1']
                    Schedule[]
                        0: CodedKern[]
```


ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

esiwace CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Summary

- Modelling required expertise in multiple disciplines (co-design)
- These disciplines work at different levels of abstraction
- Mixing science and performance can produce complex code
- Good to separate these concerns
- DSLs offer a way to do this
- DSLs support working at a high level of abstraction
- Higher level of abstraction allows a greater choice of implementation -> more performance
- Different DSLs can work at different levels of abstraction
- DSLs might support revolution and/or evolution

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988