CyML language and transpiler: Re-using biophysical processes in crop growth models across multiple platforms

Cyrille Midingoyi[1,2], Pierre Martre[1], Christophe Pradal[2,3]

[1] LEPSE, Univ Montpellier, INRAE, INSAAE, Montpellier, France (cyrille.midingoyi@inra.fr) / [2] AGAP, Univ Montpellier, CIRAD, INRAE, INSAAE, Montpellier, France / [3] Univ Montpellier, Inria, Montpellier, France

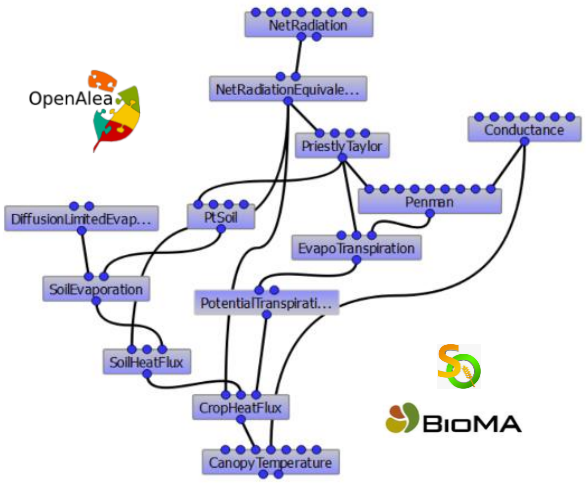
Context and Objectives

A model component implements a biophysical process in a given language within a specific simulation platform using its specific design patterns and architectural constraints. Reuse of component implements implemented in various crop simulation platforms with different programming languages is a great challenge. Instead of re-encoding and wrapping approaches to address reuse issue, we propose:

- CyML, **a minimal language** used to have an abstract representation of crop model components shared by simulation platforms through its recurrent equations or mathematical expressions with control structure;
- **a transpiler** to convert CyML models in different languages and with platforms specificities.

Approach

① Decomposition of Model component into interconnected model units



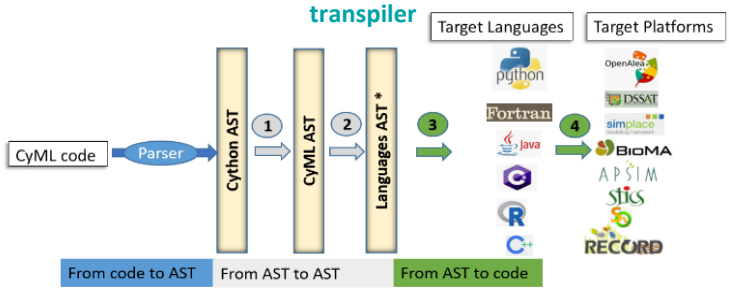
Energy Balance process decomposition

② Encoding of each unit function within CyML, a subset of Cython

CyML design through language specialization pattern

DSEL: Domain Specific Embedded Language

③ Generation of models in different languages through the transpiler



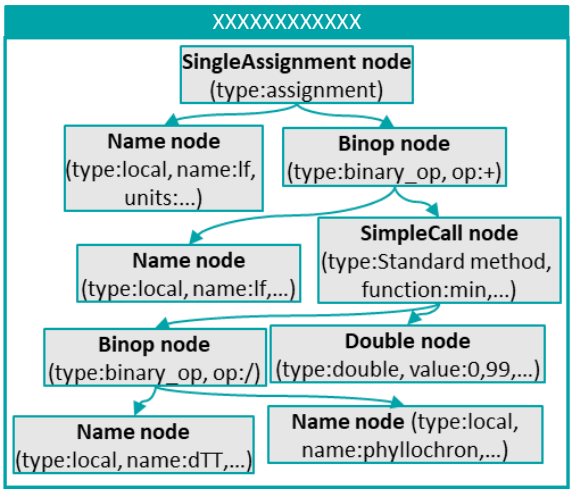
Schematic workflow of the CyML transpiler

AST: Abstract Syntax Tree

Applications

Leaf Number unit model (function)

$$lf\ (t) = lf(t-1) + \min(\frac{dTT}{phyl}, 0.99)$$



NodeVisitor

CodeGenerator

CyML model

```
def leafNumber(double lf, double dTT, double phyllochron)->int:
    """ doc
    # inputs:
        *name : lf
        *description: last leaf number
        *variablecategory: state
        * units:
        ...
    # outputs:
        * name : lf
        * description: actual leaf number
        ...
    """
    lf = lf + min(dTT/phyllochron, 0.99)
    return lf
```

F90 model

```
SUBROUTINE model_leafnumber(lf,dTT,phyllochron)
    REAL, INTENT(IN) :: dTT, phyllochron
    REAL, INTENT(INOUT) :: lf
        !- inputs:
        !       * name: lf
        !           ** description :  last leaf number
        !           ** variablecategory : state
        !...
        !- outputs:
        !       * name: lf
        !           ** description : Actual leaf number
        !           ** variablecategory : state
        !           ** datatype : DOUBLE
        lf = lf + MIN(dTT / phyllochron, 0.99)
END SUBROUTINE model_leafnumber
```

Conclusion

We demonstrate the possibility to write once a model component regardless of the simulation platform and reuse it in many platforms through transpilation. The architecture of the language and the workflow toolchain is modular allowing to extend it with new target languages and crop simulation platforms.

#DigitAg