pycropml Documentation

Release 1.2.0

Cyrille Ahmed Midingoyi

Feb 24, 2023

Contents

1 Module description 3

2 History 33

3 Credits 35

Python Module Index 37

Index 39

Contents:

CHAPTER 1

Module description

Summary

Version 1.2.0

Release 1.2.0

Date Feb 24, 2023

Author See **'authors'_** section

ChangeLog See *changelog* section

1.1 What is PyCrop2ML?

PyCrop2ML is a free, open-source library for defining and exchanging CropML models.It is used to generate components of modeling and simulation platforms from the CropML specification and allow component exchange between different platform.

It allows to parse the models described in CropML format and automatically generate the equivalent executable Python, java, C#, C++ components and packages usable from existing crop simulation platform.

1.2 What is Crop2ML ?

CropML is a XML-(JSON-)based language used to represent different biological processes involved in the crop models.

CropML project aims to provide common framework for defining and exchanging descriptions of crop growth models between crop simulation frameworks.

1.2.1 Objectives

Our main objectives are:

- define a **declarative language** to describe either an atomic model or a composition of models
- add semantic dimension to CropML language by annotation of the models to allow the composition of components of different platforms by using the standards of the semantic web
- develop a library to allow the transformation and the exchange of CropML model between different Crop modelling and simulation platform
- provide a **web repository** enabling registration, search and discovery of CropML Models
- facilitate Agricultural Model Exchange Initiative

1.2.2 Context

Nowadays, we observe the emergence of plant growth models which are built in different platforms. Although standard platform development initiatives are emerged, there is a lack of transparency, reusability, and exchange code between platforms due to the high diversity of modeling languages leading to a lack of benchmarking between the different platforms.
This project aims to gather developers and plant growth modelers to define a standard framework based on the development of declarative language and libraries to improve exchange model components between platforms.

1.2.3 Motivation

Our motivation is to:

- Strengthen the synergy between crop modelers, users and scientific researchers
- Facilitate model intercomparison (at the process level) and model improvement through the exchange of model components (algorithms) and code reuse between platforms/models.
- Bridge the gap between ecophysiologists who develop models at the process level with crop modelers and model users and facilitate the integration in crop models of new knowledge in plant science (i.e. we are seeking the exchange of knowledge rather than black box models).
- Increase capabilities and responsiveness to stakeholder' needs.
- Propose a solution to the AgMIP community for NexGen crop modeling tools.

1.2.4 Vision

- Facilitate the development of complex models
- Use modular modelling to share knowledge and rapidly develop operational tools.
- Reuse model parts to leverage the expertise of third parties;
- Renovate legacy code.
- Realize the benefit of sharing and complementing different expertise.
- Promote model sharing and reuse

1.3 CropML Description

In CropML, a model is either a model unit or a composition of models. A ModelUnit represents the atomic unit of a crop model define by the modelers. A model composition is a model resulting from the composition of two or more atomic model or composite models.

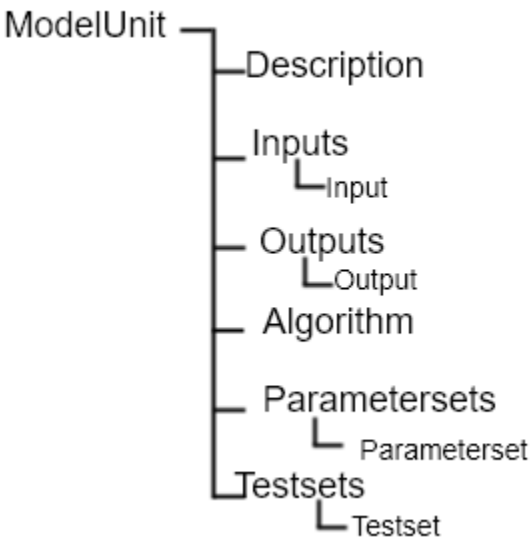
These models have a specific formal definition in CropML.

1.3.1 Formal definition of a Model Unit in CropML

The structure of a Model Unit in CropML MUST be conform to a specific Document Type Definition named ModelUnit.dtd .

So a Model Unit CropML document is a XML document well-formed and also obeys the rules given in the ModelUnit structure.

This structure MAY be described by the below tree:



Element	Description
ModelUnit	The root of an atomic model in CropML which make the difference from a composite model.
Description	some basic information related to the name of the model, its authors and others elements used to reference it.
Inputs	A list of inputs characterized by their names, initial states, the range of values and others. Its input variables are related to climate, soil and cropping system
Outputs	A list of outputs defining the processes involved, the variables whose dynamics we want to observe.
Algorithm	The description of the behaviour of the model made by the mathematical relationship between the inputs and the outputs with some control structure.
Parametersets	Some sets of parameters which are invariant and used for the simulation of the models.
Testsets	Set of model configuration used to compare estimated and desired outputs .

In the next, we define the major elements of a CropML model unit.

ModelUnit element

An atomic model in CropML is declared with <<ModelUnit>> element, the usual root of CropML ModelUnit document.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ModelUnit PUBLIC "-//SIMPLACE/DTD SOL 1.0//EN" "https://raw.
↪githubusercontent.com/AgriculturalModelExchangeInitiative/xml_representation/master/
↪ModelUnit.dtd">
<ModelUnit modelid=" " timestep=" " name=" " version="">
        ....
</ModelUnit>
```

This element MUST contain a Description, an Algorithm, Parametersets and Testsets elements and MAY optionally have Inputs and Outputs elements. The restriction of the length of different lists is not imposed.
ModelUnit element MUST have an modelid and name attributes which are used to reference an atomic model. It MUST also contain a timestep attribute to define the temporality of the model and a version attribute for each version of the model.

Description element

This element gives the general information on the model and is composed by a set of character elements. It MUST contain Title, Authors, Institution and abstract elements and MAY optionally contain URI and Reference elements.

```
<ModelUnit modelid=" " timestep=" " name=" " version =" ">
        <Description>
                <Title>title</Title>
                <Authors>authors</Authors>
                <Institution>institution</Institution>
                <URI>uri</URI>
                <Abstract><![CDATA[abstract]]></Abstract>
        </Description>
        ...
</ModelUnit>
```

Inputs elements

The inputs of Model are listed inside an XML element called Inputs within a dictionary structure composed by their attributes which declarations are optional(default, max, min, parametercategory, variablecategory and uri) or required(name, datatype, description, inputtype, unit) and their corresponding value. *Inputs* element MUST contain one or more *Input* elements.

```
<ModelUnit modelid=" " timestep=" " name=" " version =" ">
   ...
   <Inputs>
      <Input name=" " description=" " parametercategory=" " datatype=" " min=" " max=
↪" " default=" " unit=" " uri="" inputtype=" "/>
      <Input name=" " description=" " parametercategory=" " datatype=" " min=" " max=
↪" " default=" " unit=" " uri=" " inputtype=" "/>
      ...
   </Inputs>
   ...
</ModelUnit>
```

- The required *datatype* attribute is the type of input value specified in *default* (the default value in the input), *min* (the minimum value in the input) and *max* (the maximum value in the input). It MAY be one type of the set of types used in the existing crop modeling platform.
- The *inputtype* attribute makes it possible to distinguish the variables and the parameters of the model. So it MUST take one of two possible values: *parameter* and *variable*.
- The *parametercategory* attribute defines the category of parameter which is specified by one of the following values: *constant*, *species*, *soil* and *genotypic*.
- The *variablecategory* defines the category of variable depending on whether it is a *state*, a *rate* or an "auxiliary" variable. State variable characterize the behavior of the model and rate variable characterizes the changes in state variables.

Outputs element

The outputs of Model are listed inside an XML element called Outputs within a dictionary structure composed by their attributes which declarations are:

- optional(variabletype and URI)
- required(name, datatype, description, unit, max and min)

- and their corresponding value

Outputs MUST contain zero or more output elements.

```
<ModelUnit modelid=" " timestep=" " name=" " version =" ">
   ...
   <Outputs>
      <Output name=" " description=" " datatype=" " min=" " max=" "  unit=" " uri=" "/
↪>
      <Output name=" " description=" " datatype=" " min=" " max=" "  unit=" " uri=" "/
↪>
      ...
   </Outputs>
   ...
</ModelUnit>
```

The definition of different attributes is same as Input's attributes.

Algorithm element

The *Algorithm* element defines the building block of CropML model unit and shows the computational method to determine the outputs from the inputs.
It consists of a set of mathematical equations (relation between inputs), loops and conditional instructions which are well structured in a specific *language*, the algorithm's attribute.

```
<ModelUnit modelid=" " timestep=" " name=" " version =" ">
   ...
   <Algorithm language =""><![CDATA[
      ...
      ]]>
   </Algorithm>
   ...
</ModelUnit>
```

Parametersets element

Parametersets element contains one or more *Parameterset* elements that define the different ways of setting the model. Each *Parameterset* element MUST have *name* and *description* attributes that respectively represents the name and the description of each setting.

The different parameterset MUST contain a list of Param elements that show in attribute the name of the parameter (an input which inputtype equals *parameter*) and the fixed value of this one.

```
<ModelUnit modelid=" " timestep=" " name=" " version =" ">
    ...
    <Parametersets>
       <Parameterset name="" description="" uri = ""/>
       <Parameterset name="" description="" >
```

(continues on next page)

(continued from previous page)

```
          <Param name="">value</Param>
          <Param name="">value</Param>
          ...
       </Parameterset>
       ...
    ...
 </ModelUnit>
```

Testsets element

Testsets element contains one or more *Testset* elements that define the different run for evaluating the outputs of the model.

Each *Testset* element MUST have *name*, *description* and *parameterset* attributes that respectively represents the name, the description of each run and the name of the parameterset related to the Testset. This one allow to retrieve the name and the value of different parameters includes in this parameterset.

The different Testset MUST contain a list of InputValue and OutputValue elements corresponding respectively to the values of inputs used in the run and the values of Outputs that will be asserted.

```
<ModelUnit modelid=" " timestep=" " name=" " version =" ">
   ...
   <Testsets>
      <Testset name="" parameterset = "" description="" uri = ""/>
      <Testset name="" parameterset = "" description="" >
         <Test name="">
            <InputValue name="">value</InputValue>
            ...
            <OutputValue name="" precision ="">value</OutputValue>
            ...
         </Test>
         ...
      </Testset>
      ...
  </Testsets>
  ...
</ModelUnit>
```

1.3.2 Formal definition of a Composite Model in CropML

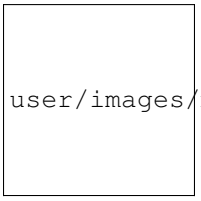
A Composite Model CropML is an assembly of processes which are described by a set of model units or a composition of models. Given a composite model is a model, this one has also inputs, outputs and internal state which describe the orchestration of different independent models composed.

The structure of a Composite Model in CropML MUST conform to a specific Document Type Definition named ModelComposition.dtd .

The composition is represented as a directed port graph of models:

Vertices are the different models that form the composition.
Ports are the inputs and outputs of each model.
Edges are directed and connect one output port to an input port of another model.

It contains in addition to all Elements of a model unit a Composition Element for the composition of models.
This structure MAY be described by the below tree:



In the next, we define the major elements of a CropML model unit.

Inputs element

It MUST contain one or more *input* element which provide a set of independent models entries. If two or more input variables of independent models are the same (same unit, interval, description) a link should be made to one input variable of the composite model.

Outputs element

It MUST contain one or more *output* element which provide a set of independent models outputs or a result of a combination of models .

Composition element

It's a list of *models* elements which contains a list of *links* elements. Link provides the mechanism for mapping inputs declared within one modelUnit to output in another modelUnit, allowing information to be exchanged between the various atomic models in the composite model.

1.3.3 Algorithm element

The implementation differs from the platform:

- Discrete Events Models and Formalisms (RECORD)
- Actor model framework (OpenAlea)
- A sequence of algorithmic instructions witch implement the control flow (BIOMA)

1.4 CyML Language Specification

This document specifies the CyML language, an extended Cython subset supported.

1.5 CyML Language Specification

This document specifies the CyML language, an extended Cython subset supported.

1.5.1 Cython file types

- The implementation files, carrying a `.pyx` suffix.

1.5.2 Basic Types

The following basic types are supported

- `bool`
- `int`
- `float`
- `double`
- `string`

1.5.3 Complex Types

The following complex types are supported

- `array`
- `list`
- `datetime`

1.5.4 Conditional Statements

- IF
- IF/Else
- IF/ElseIf/Else

The `ELIF` and `ELSE` clauses are optional. An `IF` statement can appear anywhere that a normal statement or declaration can appear

1.5.5 Integer For Loops

CyML recognises the usual Python for-in-range integer loop pattern:

```
for i in range(n):
    ...

for i in range(f,n):
    ...

for i in range(f,n, s):
    ...
```

Like other Python looping statements, break and continue may be used in the body, without else clause statement.

1.5.6 While Loop

- Like Python While loop

1.5.7 Function

- Parameters with declaration
- Default value is possible

1.5.8 Return Statement

- A function needs to return the same data type.

The following code is valid:

```
def fibonacci(int n):
    if n <= 2:
        return 1
    else:
        return fibonacci(n-1)+fibonacci(n-2)
```

1.5.9 Call functions

- Call to CyML functions are supported if the function
- is accessible on the module
- is accessible from import statement

1.5.10 Operators

Assignment

Assign	`b = a`

Unary operators

UAdd	`+a`
USub	`-a`

Binary operators

Add	`a + b`
Sub	`a - b`
Mult	`a * b`
Div	`a / b`
Pow	`a ** b`
Mod	`a % b`
BitOr	`a | b`
BitAnd	`a & b`

Augmented assign statements

AugAdd	`a += b`
AugSub	`a -= b`
AugMult	`a *= b`
AugDiv	`a /= b`

Comparison Operators

Eq	`a == b`
NotEq	`a != b`
Lt	`a < b`
LtE	`a <= b`
Gt	`a > b`
GtE	`a >= b`

Bool Operators

&&	`a and b`
||	`a or b`

1.5.11 Array routines

“ “	Return a new array of given shape and type, without initializing entries.
“ “	Return a new array of given shape and type, filled with ones.
“ “	Return a new array of given shape and type, filled with zeros.

1.5.12 Mathematical functions

Trigonometric functions

`sin(x)`	Trigonometric sine, element-wise.
`cos(x)`	Cosine elementwise.
`tan(x)`	Compute tangent element-wise.
`arcsin(x)`	Inverse sine, element-wise.
`arccos(x)`	Trigonometric inverse cosine, element-wise.
`arctan(x)`	Trigonometric inverse tangent, element-wise.

Hyperbolic functions

`sinh(x)`	Hyperbolic sine, element-wise.
`cosh(x)`	Hyperbolic cosine, element-wise.
`tanh(x)`	Compute hyperbolic tangent element-wise.

1.6 PyCrop2ML User Guide

Version 1.2.0

Release 1.2.0

Date Feb 24, 2023

This reference manual details functions, modules, and objects included in Pycrop2ML describing what they are and what they do. For a complete reference guide, see pycropml_reference.

Warning: This "Reference Guide" is still very much in progress. Many aspects of OpenAlea.Core are not covered.

1.6.1 Manual

Note: The following examples assume you have installed the packages and setup your python path correctly.

Installation

```
conda install -c amei -c conda-forge pycropml
```

or

```
python setup.py install
```

Overview of the different classes

1.7 src

1.7.1 pycropml package

Subpackages

pycropml.interface package

Submodules

pycropml.interface.design module

pycropml.interface.version module

Module contents

pycropml.transpiler package

Subpackages

pycropml.transpiler.antlr_py package

Subpackages

pycropml.transpiler.antlr_py.bioma package

Submodules

pycropml.transpiler.antlr_py.bioma.biomaExtraction module

pycropml.transpiler.antlr_py.bioma.run module

pycropml.transpiler.antlr_py.bioma.run2 module

Module contents

pycropml.transpiler.antlr_py.cmake package

Submodules

pycropml.transpiler.antlr_py.cmake.cmakeTransformer module

pycropml.transpiler.antlr_py.cmake.cmake_generate_tree module

pycropml.transpiler.antlr_py.cmake.cmake_generate_tree.**generate**(*parser*)

Module contents

pycropml.transpiler.antlr_py.csharp package

Submodules

pycropml.transpiler.antlr_py.csharp.api_transform module

pycropml.transpiler.antlr_py.csharp.builtin_typed_api module

pycropml.transpiler.antlr_py.csharp.cs_cyml module

pycropml.transpiler.antlr_py.csharp.csharpRules module

pycropml.transpiler.antlr_py.csharp.csharpTransformer module

pycropml.transpiler.antlr_py.csharp.csharp_generate_tree module

pycropml.transpiler.antlr_py.csharp.csharp_generate_tree.**generate**(*parser*)

pycropml.transpiler.antlr_py.csharp.csharp_preprocessing module

Module contents

pycropml.transpiler.antlr_py.dssat package

Submodules

pycropml.transpiler.antlr_py.dssat.dssatExtraction module

pycropml.transpiler.antlr_py.dssat.dssatExtraction2 module

pycropml.transpiler.antlr_py.dssat.run module

Module contents

pycropml.transpiler.antlr_py.fortran package

Submodules

pycropml.transpiler.antlr_py.fortran.api_transform module

pycropml.transpiler.antlr_py.fortran.builtin_typed_api module

pycropml.transpiler.antlr_py.fortran.f90_cyml module

pycropml.transpiler.antlr_py.fortran.fortranExtraction module

pycropml.transpiler.antlr_py.fortran.fortranTransformer module

pycropml.transpiler.antlr_py.fortran.fortran_generate_tree module

pycropml.transpiler.antlr_py.fortran.fortran_generate_tree.**generate**(*parser*)

pycropml.transpiler.antlr_py.fortran.fortran_preprocessing module

pycropml.transpiler.antlr_py.fortran.run module

Module contents

pycropml.transpiler.antlr_py.grammars package

Submodules

pycropml.transpiler.antlr_py.grammars.CMakeLexer module

pycropml.transpiler.antlr_py.grammars.CMakeListener module

pycropml.transpiler.antlr_py.grammars.CMakeParser module

pycropml.transpiler.antlr_py.grammars.CMakeVisitor module

pycropml.transpiler.antlr_py.grammars.CPP14Lexer module

pycropml.transpiler.antlr_py.grammars.CPP14Listener module

pycropml.transpiler.antlr_py.grammars.CPP14Parser module

pycropml.transpiler.antlr_py.grammars.CPP14Visitor module

pycropml.transpiler.antlr_py.grammars.CSharpLexer module

pycropml.transpiler.antlr_py.grammars.CSharpParser module

pycropml.transpiler.antlr_py.grammars.CSharpParser3 module

pycropml.transpiler.antlr_py.grammars.CSharpParserListener module

pycropml.transpiler.antlr_py.grammars.CSharpParserListener2 module

pycropml.transpiler.antlr_py.grammars.CSharpParserVisitor module

pycropml.transpiler.antlr_py.grammars.CommentsLexer module

pycropml.transpiler.antlr_py.grammars.CommentsListener module

pycropml.transpiler.antlr_py.grammars.CommentsParser module

pycropml.transpiler.antlr_py.grammars.CommentsVisitor module

pycropml.transpiler.antlr_py.grammars.Fortran77Lexer module

pycropml.transpiler.antlr_py.grammars.Fortran77Parser module

pycropml.transpiler.antlr_py.grammars.Fortran77ParserListener module

pycropml.transpiler.antlr_py.grammars.Fortran77ParserVisitor module

pycropml.transpiler.antlr_py.grammars.Fortran90Lexer module

pycropml.transpiler.antlr_py.grammars.Fortran90Parser module

pycropml.transpiler.antlr_py.grammars.Fortran90ParserListener module

pycropml.transpiler.antlr_py.grammars.Fortran90ParserVisitor module

pycropml.transpiler.antlr_py.grammars.Java8Lexer module

pycropml.transpiler.antlr_py.grammars.Java8Parser module

pycropml.transpiler.antlr_py.grammars.Java8ParserListener module

pycropml.transpiler.antlr_py.grammars.Java8ParserVisitor module

pycropml.transpiler.antlr_py.grammars.PythonLexer module

pycropml.transpiler.antlr_py.grammars.PythonLexerBase module

pycropml.transpiler.antlr_py.grammars.PythonParser module

pycropml.transpiler.antlr_py.grammars.PythonParserBase module

pycropml.transpiler.antlr_py.grammars.PythonParserListener module

pycropml.transpiler.antlr_py.grammars.PythonParserVisitor module

pycropml.transpiler.antlr_py.grammars.RLexer module

pycropml.transpiler.antlr_py.grammars.RListener module

pycropml.transpiler.antlr_py.grammars.RParser module

pycropml.transpiler.antlr_py.grammars.RVisitor module

pycropml.transpiler.antlr_py.grammars.XMLLexer module

pycropml.transpiler.antlr_py.grammars.XMLParser module

pycropml.transpiler.antlr_py.grammars.XMLParserListener module

pycropml.transpiler.antlr_py.grammars.XMLParserVisitor module

Module contents

pycropml.transpiler.antlr_py.java package

Submodules

pycropml.transpiler.antlr_py.java.api_transform module

pycropml.transpiler.antlr_py.java.builtin_typed_api module

pycropml.transpiler.antlr_py.java.javaRules module

pycropml.transpiler.antlr_py.java.javaTransformer module

pycropml.transpiler.antlr_py.java.java_cyml module

pycropml.transpiler.antlr_py.java.java_generate_tree module

pycropml.transpiler.antlr_py.java.java_generate_tree.**generate**(*parser*)

pycropml.transpiler.antlr_py.java.java_preprocessing module

Module contents

pycropml.transpiler.antlr_py.openalea package

Submodules

pycropml.transpiler.antlr_py.openalea.openaleaExtraction module

pycropml.transpiler.antlr_py.openalea.run module

Module contents

pycropml.transpiler.antlr_py.python package

Submodules

pycropml.transpiler.antlr_py.python.api_transform module

pycropml.transpiler.antlr_py.python.builtin_typed_api module

pycropml.transpiler.antlr_py.python.pythonTransformer module

pycropml.transpiler.antlr_py.python.python_generate_tree module

pycropml.transpiler.antlr_py.python.python_generate_tree.**generate**(*parser*)

pycropml.transpiler.antlr_py.python.python_preprocessing module

Module contents

pycropml.transpiler.antlr_py.simplace package

Submodules

pycropml.transpiler.antlr_py.simplace.run module

pycropml.transpiler.antlr_py.simplace.simplaceExtraction module

Module contents

pycropml.transpiler.antlr_py.stics package

Submodules

pycropml.transpiler.antlr_py.stics.read_routines module

pycropml.transpiler.antlr_py.stics.run module

pycropml.transpiler.antlr_py.stics.sticsExtraction module

Module contents

pycropml.transpiler.antlr_py.tests package

Submodules

pycropml.transpiler.antlr_py.tests.test_bioma module

pycropml.transpiler.antlr_py.tests.test_cs module

pycropml.transpiler.antlr_py.tests.test_dssat module

pycropml.transpiler.antlr_py.tests.test_dssat2 module

pycropml.transpiler.antlr_py.tests.test_openalea module

pycropml.transpiler.antlr_py.tests.test_simplace module

pycropml.transpiler.antlr_py.tests.test_to_specification module

Module contents

pycropml.transpiler.antlr_py.xml package

Submodules

pycropml.transpiler.antlr_py.xml.xmlTransformer module

pycropml.transpiler.antlr_py.xml.xml_generate_tree module

pycropml.transpiler.antlr_py.xml.xml_generate_tree.**generate**(*parser*)

Module contents

Submodules

pycropml.transpiler.antlr_py.api_declarations module

pycropml.transpiler.antlr_py.codeExtraction module

pycropml.transpiler.antlr_py.codeExtraction.**extraction**(*text_*, *balise_start*, *balise_end*, *ignore_start=None*, *ignore_end=None*)

pycropml.transpiler.antlr_py.codeExtraction.**remove**(*text_*, *ignore_start*, *ignore_end*)

pycropml.transpiler.antlr_py.createXml module

pycropml.transpiler.antlr_py.extract_metadata module

pycropml.transpiler.antlr_py.extract_metadata_from_comment module

pycropml.transpiler.antlr_py.extraction module

pycropml.transpiler.antlr_py.extraction.**ExtractComments**(*all_text*, *c_st_single*, *c_st_multi*, *c_end_multi*, *pos=None*)

pycropml.transpiler.antlr_py.extraction.**lineNumber**(*s*, *l*)

pycropml.transpiler.antlr_py.extraction.**multiLineNumber**(*vn*, *zz*)

pycropml.transpiler.antlr_py.extraction.**replace**(*s*, *l*)

pycropml.transpiler.antlr_py.generateCyml module

pycropml.transpiler.antlr_py.listOfTags module

pycropml.transpiler.antlr_py.parse module

pycropml.transpiler.antlr_py.repowalk module

pycropml.transpiler.antlr_py.repowalk.**walk**(*path*, *ext=None*)

pycropml.transpiler.antlr_py.simplifyAntlrTree module

pycropml.transpiler.antlr_py.to_CASG module

pycropml.transpiler.antlr_py.to_specification module

pycropml.transpiler.antlr_py.toxml module

module for generating and serializing xml and html structures by using simple python objects.

(c) holger krekel, holger at merlinux eu. 2009

from https://raw.githubusercontent.com/pytest-dev/py/1805da33af0fdc89d7f00dd14e3a5f567c374e13/py/_xmlgen.py

class pycropml.transpiler.antlr_py.toxml.**HtmlTag**(**args*, ***kwargs*)
Bases: *pycropml.transpiler.antlr_py.toxml.Tag*

unicode(*indent=2*)

class pycropml.transpiler.antlr_py.toxml.**HtmlVisitor**(*write*, *indent=0*, *curindent=0*, *shortempty=True*)
Bases: *pycropml.transpiler.antlr_py.toxml.SimpleUnicodeVisitor*

inline = {'a': 1, 'abbr': 1, 'acronym': 1, 'b': 1, 'basefont': 1, 'bdo': 1, 'big

repr_attribute(*attrs*, *name*)

single = {'area': 1, 'base': 1, 'br': 1, 'col': 1, 'frame': 1, 'hr': 1, 'img':

x = 'var'

class pycropml.transpiler.antlr_py.toxml.**Namespace**
Bases: object

class pycropml.transpiler.antlr_py.toxml.**NamespaceMetaclass**
Bases: type

class pycropml.transpiler.antlr_py.toxml.**SimpleUnicodeVisitor**(*write*, *indent=0*, *curindent=0*, *shortempty=True*)
Bases: object

recursive visitor to write unicode.

Tag(*tag*)

attributes(*tag*)

getstyle(*tag*)
return attribute list suitable for styling.

list(*obj*)

raw(*obj*)

repr_attribute(*attrs*, *name*)

visit(*node*)
dispatcher on node's class/bases name.

class pycropml.transpiler.antlr_py.toxml.**Tag**(**args*, ***kwargs*)
Bases: list

class Attr(***kwargs*)
Bases: object

unicode(*indent=2*)

class pycropml.transpiler.antlr_py.toxml.**html**
Bases: *pycropml.transpiler.antlr_py.toxml.Namespace*

class Style(***kw*)
Bases: object

x = 'wbr'

class pycropml.transpiler.antlr_py.toxml.**raw**(*uniobj*)
Bases: object

just a box that can contain a unicode string that will be included directly in the output

pycropml.transpiler.antlr_py.toxml.**u**(*s*)

Module contents

pycropml.transpiler.generators package

Submodules

pycropml.transpiler.generators.apsimGenerator module

pycropml.transpiler.generators.biomaGenerator module

pycropml.transpiler.generators.biomaGenerator2 module

pycropml.transpiler.generators.checkGenerator module

pycropml.transpiler.generators.cppGenerator module

pycropml.transpiler.generators.csharpGenerator module

pycropml.transpiler.generators.cymlGenerator module

pycropml.transpiler.generators.docGenerator module

pycropml.transpiler.generators.dssatGenerator module

pycropml.transpiler.generators.fortranGenerator module

pycropml.transpiler.generators.javaGenerator module

pycropml.transpiler.generators.openaleaGenerator module

pycropml.transpiler.generators.pythonGenerator module

pycropml.transpiler.generators.rGenerator module

pycropml.transpiler.generators.recordGenerator module

pycropml.transpiler.generators.simplaceGenerator module

pycropml.transpiler.generators.sirius2Generator module

pycropml.transpiler.generators.siriusGenerator module

pycropml.transpiler.generators.sticsGenerator module

Module contents

pycropml.transpiler.lib package

Module contents

pycropml.transpiler.rules package

Submodules

pycropml.transpiler.rules.cppRules module

pycropml.transpiler.rules.csharpRules module

pycropml.transpiler.rules.cymlRules module

pycropml.transpiler.rules.fortranRules module

pycropml.transpiler.rules.generalRule module

class pycropml.transpiler.rules.generalRule.**GeneralRule**
Bases: object

” Abstract class of Rules

pycropml.transpiler.rules.javaRules module

pycropml.transpiler.rules.pythonRules module

pycropml.transpiler.rules.rRules module

pycropml.transpiler.rules.sqRules module

Module contents

Submodules

pycropml.transpiler.Parser module

pycropml.transpiler.api_transform module

pycropml.transpiler.ast_transform module

pycropml.transpiler.builtin_typed_api module

pycropml.transpiler.checkingModel module

class pycropml.transpiler.checkingModel.**Checking**
Module used to check units validity in model equation based on model xml files. This checking can also use for python code with metadata

pycropml.transpiler.codeGenerator module

pycropml.transpiler.env module

class pycropml.transpiler.env.**Env**(*values=None*, *parent=None*)

child_env(*values=None*)

pycropml.transpiler.errors module

exception pycropml.transpiler.errors.**PseudoCythonNotTranslatableError**(*message*, *suggestions=None*, *right=None*, *wrong=None*)
Bases: *pycropml.transpiler.errors.PseudoError*

exception pycropml.transpiler.errors.**PseudoCythonTypeCheckError**(*message*, *suggestions=None*, *right=None*, *wrong=None*)

Bases: *pycropml.transpiler.errors.PseudoError*

exception pycropml.transpiler.errors.**PseudoError**(*message*, *suggestions=None*, *right=None*, *wrong=None*)

Bases: exceptions.Exception

pycropml.transpiler.errors.**beautiful_error**(*exception*)

pycropml.transpiler.errors.**cant_infer_error**(*name*, *line*)

pycropml.transpiler.errors.**isNotImplemented**(*data*, *location*)

pycropml.transpiler.errors.**notInmplemented**(*exception*)

pycropml.transpiler.errors.**tab_aware**(*location*, *code*)
if tabs in beginning of code, add tabs for them, otherwise spaces

pycropml.transpiler.errors.**translation_error**(*data*, *location=None*, *code=None*, *wrong_type=None*, ***options*)

pycropml.transpiler.errors.**type_check_error**(*data*, *location=None*, *code=None*, *wrong_type=None*, ***options*)

pycropml.transpiler.helpers module

pycropml.transpiler.helpers.**prepare_table**(*types*, *original_methods=None*)

pycropml.transpiler.helpers.**safe_serialize_type**(*l*)
serialize only with letters, numbers and _

pycropml.transpiler.helpers.**serialize_type**(*l*)

pycropml.transpiler.interface module

pycropml.transpiler.main module

pycropml.transpiler.nodeVisitor module

pycropml.transpiler.preprocessing module

pycropml.transpiler.pseudo_tree module

pycropml.transpiler.version module

Maintain version for this package. Do not edit this file, use 'version' section of config.

pycropml.transpiler.version.**MAJOR = 0**
(int) Version major component.

pycropml.transpiler.version.**MINOR = 0**
(int) Version minor component.

pycropml.transpiler.version.**POST = 2**
(int) Version post or bugfix component.

Module contents

pycropml.units package

Submodules

pycropml.units.registry module

pycropml.units.unit_object module

Module contents

Submodules

pycropml.algorithm module

class pycropml.algorithm.**Algorithm**(*language*, *development*, *platform*, *filename=None*)
Bases: object

pycropml.checking module

class pycropml.checking.**Test**(*name*)
Bases: *pycropml.checking.Testset*

class pycropml.checking.**Testset**(*name*, *parameterset*, *description*, *uri=None*)
Bases: object

Test

pycropml.checking.**testset**(*model*, *name*, *kwds*)

pycropml.code2nbk module

pycropml.composition module

pycropml.cyml module

pycropml.description module

class pycropml.description.**Description**
Bases: object

Model Unit Description.

A description is defined by:

- Title

- Authors
- Institution
- Reference
- Abstract

pycropml.error module

Created on Wed Apr 10 17:01:34 2019

@author: midingoy

exception pycropml.error.**Error**(*message*)
Bases: exceptions.Exception

pycropml.formater_f90 module

pycropml.formater_f90.**formater**(*code*)

pycropml.formater_f90.**formaterNext**(*line*)

pycropml.function module

class pycropml.function.**Function**(*name*, *language*, *filename*, *type*, *description*)
Bases: object

pycropml.initialization module

class pycropml.initialization.**Initialization**(*name*, *language*, *filename*)
Bases: object

Function

pycropml.inout module

class pycropml.inout.**Input**(*kwds*)
Bases: *pycropml.inout.InputOutput*

class pycropml.inout.**InputOutput**(*kwds*)
Bases: object

class pycropml.inout.**Output**(*kwds*)
Bases: *pycropml.inout.InputOutput*

pycropml.main module

pycropml.model module

Created on Tue Jun 4 22:10:54 2019

@author: midingoy

pycropml.model.**module_exists**(*module_name*)

pycropml.modelunit module

Model Description and Model Unit.

class pycropml.modelunit.**ModelDefinition**(*kwds*)
Bases: object

class pycropml.modelunit.**ModelUnit**(*kwds*)
Bases: pycropml.modelunit.ModelDefinition

Formal description of a Model Unit.

add_description(*description*)
TODO

auxiliary

exogenous

parameters

rates

states

pycropml.nameconvention module

pycropml.nameconvention.**signature**(*model*, *format*)
summary

Parameters **model** (*ModelUnit*) – A Python object of a Crop2ML model Unit

Returns name

Return type str

pycropml.nameconvention.**signature1**(*model*)
summary

Parameters **model** (*ModelUnit*) – A Python object of a Crop2ML model Unit

Returns name

Return type str

pycropml.nameconvention.**signature2**(*model*)
summary

Parameters **model** (*ModelUnit*) – A Python object of a Crop2ML model Unit

Returns name

Return type str

pycropml.package module

pycropml.parameterset module

class pycropml.parameterset.**Parameterset**(*name*, *description*, *uri=None*)
Bases: object

Parameter set

pycropml.parameterset.**parameterset**(*model*, *name*, *kwds*)

pycropml.pparse module

pycropml.render_cpp module

pycropml.render_csharp module

pycropml.render_cyml module

pycropml.render_fortran module

pycropml.render_java module

pycropml.render_notebook module

pycropml.render_notebook_csharp module

pycropml.render_notebook_java module

pycropml.render_python module

pycropml.render_r module

pycropml.test_generator module

pycropml.topology module

pycropml.unitparser module

pycropml.version module

Maintain version for this package. Do not edit this file, use 'version' section of config.

pycropml.version.**MAJOR = 1**
(int) Version major component.

pycropml.version.**MINOR = 2**
(int) Version minor component.

pycropml.version.**POST = 0**
(int) Version post or bugfix component.

pycropml.writeTest module

pycropml.xml2wf module

Module contents

1.8 Usecases

1.9 Licence

PyCropML is released under a MIT License.

1.10 Contributing Guide

This is a wiki for anything related to the contributing on Crop2ML which is a project of the Agricultural Model Exchange Initiative. For more information about this project, please visit CropML documentation Crop2ML documentation.

1.11 Quick Links

- Learn how to contribute to Crop2ML.
- Learn the architecture of Pycrop2ml.

1.12 Publications

1.13 Glossary

Terminology

Model Simplified representation of the crop system within specific objectives.

1.14 History

1.14.1 PyCrop2ML creation (2018-01-18)

- First release on Conda.

Overview

PyCrop2ML is able to:

- Manage Crop2ML packages and model components
- Transform them to crop modeling and simulation platforms
- Visualize composite models

- A PDF version of PyCrop2ML documentation is available.

CHAPTER 2

History

2.1 PyCrop2ML creation (2018-01-18)

- First release on Conda.

CHAPTER 3

Credits

1. Development Lead

- Cyrille Ahmed Midingoyi, <cyrille.midingoyi@inra.fr>
- Christophe Pradal, <christophe.pradal@cirad.fr>

2. Contributors

None yet. Why not be the first?

3.1 Indices and tables

- genindex
- modindex
- search

3.2 License

PyCrop2ML is released under a MIT License.

3.3 Supported by:


RECORD


BioMA


images/siriusquality.png


images/simplace.png

Python Module Index

p

pycropml, 1
pycropml.algorithm, 27
pycropml.checking, 27
pycropml.description, 27
pycropml.error, 28
pycropml.formater_f90, 28
pycropml.function, 28
pycropml.initialization, 28
pycropml.inout, 28
pycropml.model, 28
pycropml.nameconvention, 29
pycropml.parameterset, 30
pycropml.transpiler, 27
pycropml.transpiler.antlr_py, 23
pycropml.transpiler.antlr_py.bioma, 15
pycropml.transpiler.antlr_py.cmake, 15
pycropml.transpiler.antlr_py.cmake.cmake_generate_tree, 15
pycropml.transpiler.antlr_py.codeExtraction, 21
pycropml.transpiler.antlr_py.csharp, 16
pycropml.transpiler.antlr_py.csharp.csharp_generate_tree, 16
pycropml.transpiler.antlr_py.dssat, 16
pycropml.transpiler.antlr_py.extraction, 21
pycropml.transpiler.antlr_py.fortran, 17
pycropml.transpiler.antlr_py.fortran.fortran_generate_tree, 16
pycropml.transpiler.antlr_py.java, 19
pycropml.transpiler.antlr_py.java.java_generate_tree, 19
pycropml.transpiler.antlr_py.listOfTags, 22
pycropml.transpiler.antlr_py.openalea, 19
pycropml.transpiler.antlr_py.python, 20
pycropml.transpiler.antlr_py.python.python_generate 20
pycropml.transpiler.antlr_py.repowalk, 22
pycropml.transpiler.antlr_py.simplace, 20
pycropml.transpiler.antlr_py.stics, 20
pycropml.transpiler.antlr_py.tests, 21
pycropml.transpiler.antlr_py.toxml, 22
pycropml.transpiler.antlr_py.xml, 21
pycropml.transpiler.antlr_py.xml.xml_generate_tree, 21
pycropml.transpiler.checkingModel, 25
pycropml.transpiler.env, 25
pycropml.transpiler.errors, 25
pycropml.transpiler.generators, 24
pycropml.transpiler.helpers, 26
pycropml.transpiler.lib, 24
pycropml.transpiler.rules, 25
pycropml.transpiler.rules.generalRule, 24
pycropml.transpiler.version, 26
pycropml.version, 30

Index

A

add_description() (*pycropml.modelunit.ModelUnit method*), 29
Algorithm (*class in pycropml.algorithm*), 27
attributes() (*pycropml.transpiler.antlr_py.toxml.SimpleUnicodeVisitor method*), 22
auxiliary (*pycropml.modelunit.ModelUnit attribute*), 29

B

beautiful_error() (*in module pycropml.transpiler.errors*), 26

C

cant_infer_error() (*in module pycropml.transpiler.errors*), 26
Checking (*class in pycropml.transpiler.checkingModel*), 25
child_env() (*pycropml.transpiler.env.Env method*), 25

D

Description (*class in pycropml.description*), 27

E

Env (*class in pycropml.transpiler.env*), 25
Error, 28
exogenous (*pycropml.modelunit.ModelUnit attribute*), 29
ExtractComments() (*in module pycropml.transpiler.antlr_py.extraction*), 21
extraction() (*in module pycropml.transpiler.antlr_py.codeExtraction*), 21

F

formater() (*in module pycropml.formater_f90*), 28
formaterNext() (*in module pycropml.formater_f90*), 28
Function (*class in pycropml.function*), 28

G

GeneralRule (*class in pycropml.transpiler.rules.generalRule*), 24
generate() (*in module pycropml.transpiler.antlr_py.cmake.cmake_generate_tree*), 15
generate() (*in module pycropml.transpiler.antlr_py.csharp.csharp_generate_tree*), 16
generate() (*in module pycropml.transpiler.antlr_py.fortran.fortran_generate_tree*), 16
generate() (*in module pycropml.transpiler.antlr_py.java.java_generate_tree*), 19
generate() (*in module pycropml.transpiler.antlr_py.python.python_generate_tree*), 20
generate() (*in module pycropml.transpiler.antlr_py.xml.xml_generate_tree*), 21
getstyle() (*pycropml.transpiler.antlr_py.toxml.SimpleUnicodeVisitor method*), 23

H

html (*class in pycropml.transpiler.antlr_py.toxml*), 23
html.Style (*class in pycropml.transpiler.antlr_py.toxml*), 23
HtmlTag (*class in pycropml.transpiler.antlr_py.toxml*), 22
HtmlVisitor (*class in pycropml.transpiler.antlr_py.toxml*), 22

I

Initialization (*class in pycropml.initialization*), 28
inline (*pycropml.transpiler.antlr_py.toxml.HtmlVisitor attribute*), 22

Input (*class in pycropml.inout*), 28
InputOutput (*class in pycropml.inout*), 28
isNotImplemented() (*in module pycropml.transpiler.errors*), 26

L

lineNumber() (*in module pycropml.transpiler.antlr_py.extraction*), 21
list() (*pycropml.transpiler.antlr_py.toxml.SimpleUnicodeVisitor method*), 23

M

MAJOR (*in module pycropml.transpiler.version*), 26
MAJOR (*in module pycropml.version*), 30
MINOR (*in module pycropml.transpiler.version*), 26
MINOR (*in module pycropml.version*), 30
Model, **31**
ModelDefinition (*class in pycropml.modelunit*), 29
ModelUnit (*class in pycropml.modelunit*), 29
module_exists() (*in module pycropml.model*), 28
multiLineNumber() (*in module pycropml.transpiler.antlr_py.extraction*), 21

N

Namespace (*class in pycropml.transpiler.antlr_py.toxml*), 22
NamespaceMetaclass (*class in pycropml.transpiler.antlr_py.toxml*), 22
notInmplemented() (*in module pycropml.transpiler.errors*), 26

O

Output (*class in pycropml.inout*), 28

P

parameters (*pycropml.modelunit.ModelUnit attribute*), 29
Parameterset (*class in pycropml.parameterset*), 30
parameterset() (*in module pycropml.parameterset*), 30
POST (*in module pycropml.transpiler.version*), 26
POST (*in module pycropml.version*), 30
prepare_table() (*in module pycropml.transpiler.helpers*), 26
PseudoCythonNotTranslatableError, 25
PseudoCythonTypeCheckError, 25
PseudoError, 26
pycropml (*module*), 1, 31
pycropml.algorithm (*module*), 27
pycropml.checking (*module*), 27
pycropml.description (*module*), 27
pycropml.error (*module*), 28
pycropml.formater_f90 (*module*), 28
pycropml.function (*module*), 28
pycropml.initialization (*module*), 28
pycropml.inout (*module*), 28
pycropml.model (*module*), 28
pycropml.modelunit (*module*), 29
pycropml.nameconvention (*module*), 29
pycropml.parameterset (*module*), 30
pycropml.transpiler (*module*), 27
pycropml.transpiler.antlr_py (*module*), 23
pycropml.transpiler.antlr_py.bioma (*module*), 15
pycropml.transpiler.antlr_py.cmake (*module*), 15
pycropml.transpiler.antlr_py.cmake.cmake_generate_t (*module*), 15
pycropml.transpiler.antlr_py.codeExtraction (*module*), 21
pycropml.transpiler.antlr_py.csharp (*module*), 16
pycropml.transpiler.antlr_py.csharp.csharp_generate (*module*), 16
pycropml.transpiler.antlr_py.dssat (*module*), 16
pycropml.transpiler.antlr_py.extraction (*module*), 21
pycropml.transpiler.antlr_py.fortran (*module*), 17
pycropml.transpiler.antlr_py.fortran.fortran_genera (*module*), 16
pycropml.transpiler.antlr_py.java (*module*), 19
pycropml.transpiler.antlr_py.java.java_generate_tre (*module*), 19
pycropml.transpiler.antlr_py.listOfTags (*module*), 22
pycropml.transpiler.antlr_py.openalea (*module*), 19
pycropml.transpiler.antlr_py.python (*module*), 20
pycropml.transpiler.antlr_py.python.python_generate (*module*), 20
pycropml.transpiler.antlr_py.repowalk (*module*), 22
pycropml.transpiler.antlr_py.simplace (*module*), 20
pycropml.transpiler.antlr_py.stics (*module*), 20
pycropml.transpiler.antlr_py.tests (*module*), 21
pycropml.transpiler.antlr_py.toxml (*module*), 22
pycropml.transpiler.antlr_py.xml (*module*), 21
pycropml.transpiler.antlr_py.xml.xml_generate_tree

(*module*), 21
pycropml.transpiler.checkingModel (*module*), 25
pycropml.transpiler.env (*module*), 25
pycropml.transpiler.errors (*module*), 25
pycropml.transpiler.generators (*module*), 24
pycropml.transpiler.helpers (*module*), 26
pycropml.transpiler.lib (*module*), 24
pycropml.transpiler.rules (*module*), 25
pycropml.transpiler.rules.generalRule (*module*), 24
pycropml.transpiler.version (*module*), 26
pycropml.version (*module*), 30

R

rates (*pycropml.modelunit.ModelUnit attribute*), 29
raw (*class in pycropml.transpiler.antlr_py.toxml*), 23
raw() (*pycropml.transpiler.antlr_py.toxml.SimpleUnicodeVisitor method*), 23
remove() (*in module pycropml.transpiler.antlr_py.codeExtraction*), 21
replace() (*in module pycropml.transpiler.antlr_py.extraction*), 21
repr_attribute() (*pycropml.transpiler.antlr_py.toxml.HtmlVisitor method*), 22
repr_attribute() (*pycropml.transpiler.antlr_py.toxml.SimpleUnicodeVisitor method*), 23

S

safe_serialize_type() (*in module pycropml.transpiler.helpers*), 26
serialize_type() (*in module pycropml.transpiler.helpers*), 26
signature() (*in module pycropml.nameconvention*), 29
signature1() (*in module pycropml.nameconvention*), 29
signature2() (*in module pycropml.nameconvention*), 29
SimpleUnicodeVisitor (*class in pycropml.transpiler.antlr_py.toxml*), 22
single (*pycropml.transpiler.antlr_py.toxml.HtmlVisitor attribute*), 22
states (*pycropml.modelunit.ModelUnit attribute*), 29

T

tab_aware() (*in module pycropml.transpiler.errors*), 26
Tag (*class in pycropml.transpiler.antlr_py.toxml*), 23
Tag() (*pycropml.transpiler.antlr_py.toxml.SimpleUnicodeVisitor method*), 22
Tag.Attr (*class in pycropml.transpiler.antlr_py.toxml*), 23
Test (*class in pycropml.checking*), 27
Testset (*class in pycropml.checking*), 27
testset() (*in module pycropml.checking*), 27
translation_error() (*in module pycropml.transpiler.errors*), 26
type_check_error() (*in module pycropml.transpiler.errors*), 26

U

u() (*in module pycropml.transpiler.antlr_py.toxml*), 23
unicode() (*pycropml.transpiler.antlr_py.toxml.HtmlTag method*), 22
unicode() (*pycropml.transpiler.antlr_py.toxml.Tag method*), 23

V

visit() (*pycropml.transpiler.antlr_py.toxml.SimpleUnicodeVisitor method*), 23

W

walk() (*in module pycropml.transpiler.antlr_py.repowalk*), 22

X

x (*pycropml.transpiler.antlr_py.toxml.html attribute*), 23
x (*pycropml.transpiler.antlr_py.toxml.HtmlVisitor attribute*), 22