MEAGAN LANG

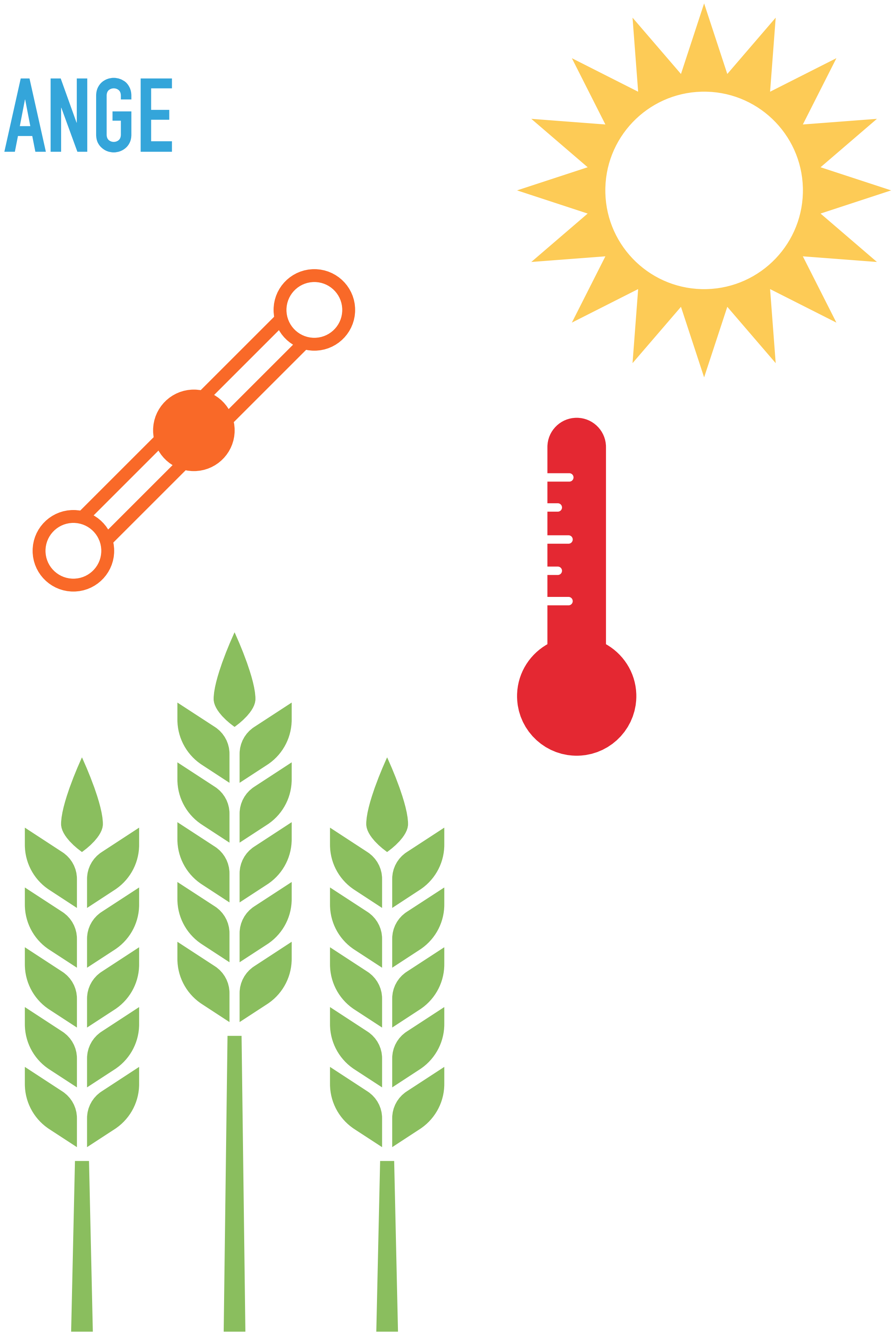
CIS_INTERFACE:

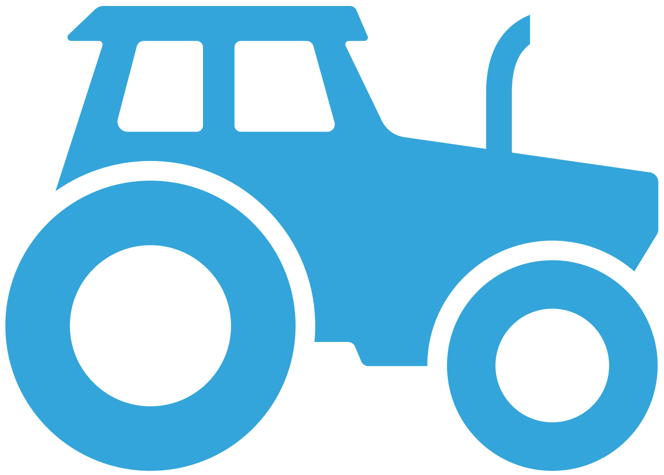
A PYTHON PACKAGE FOR CONNECTING SCIENTIFIC MODELS ACROSS SCALES AND LANGUAGES

BACKGROUND

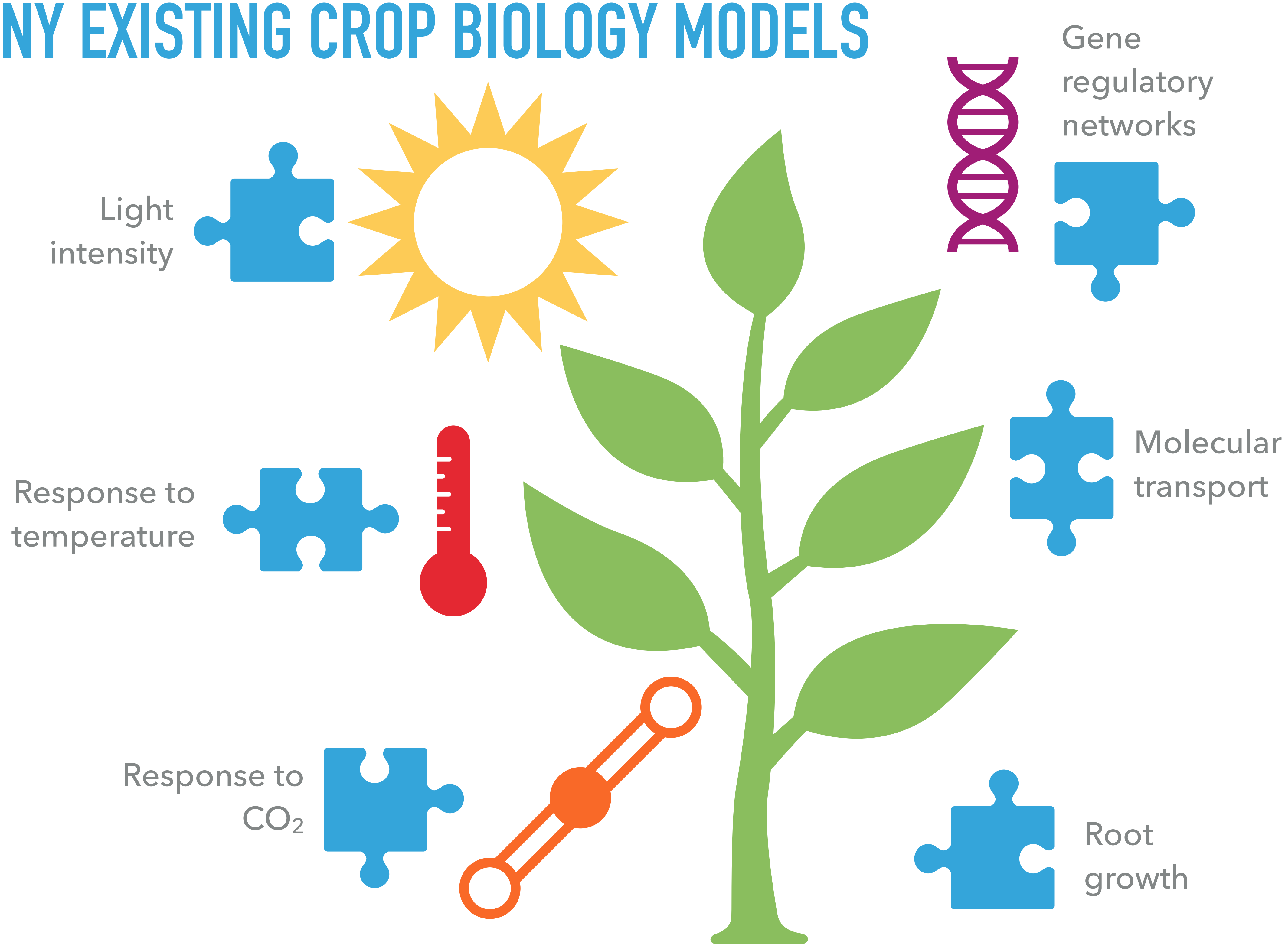
FOOD SECURITY UNDER CLIMATE CHANGE

With increasing temperatures and CO_2 levels, are there actions that can be taken to ensure there is enough food?



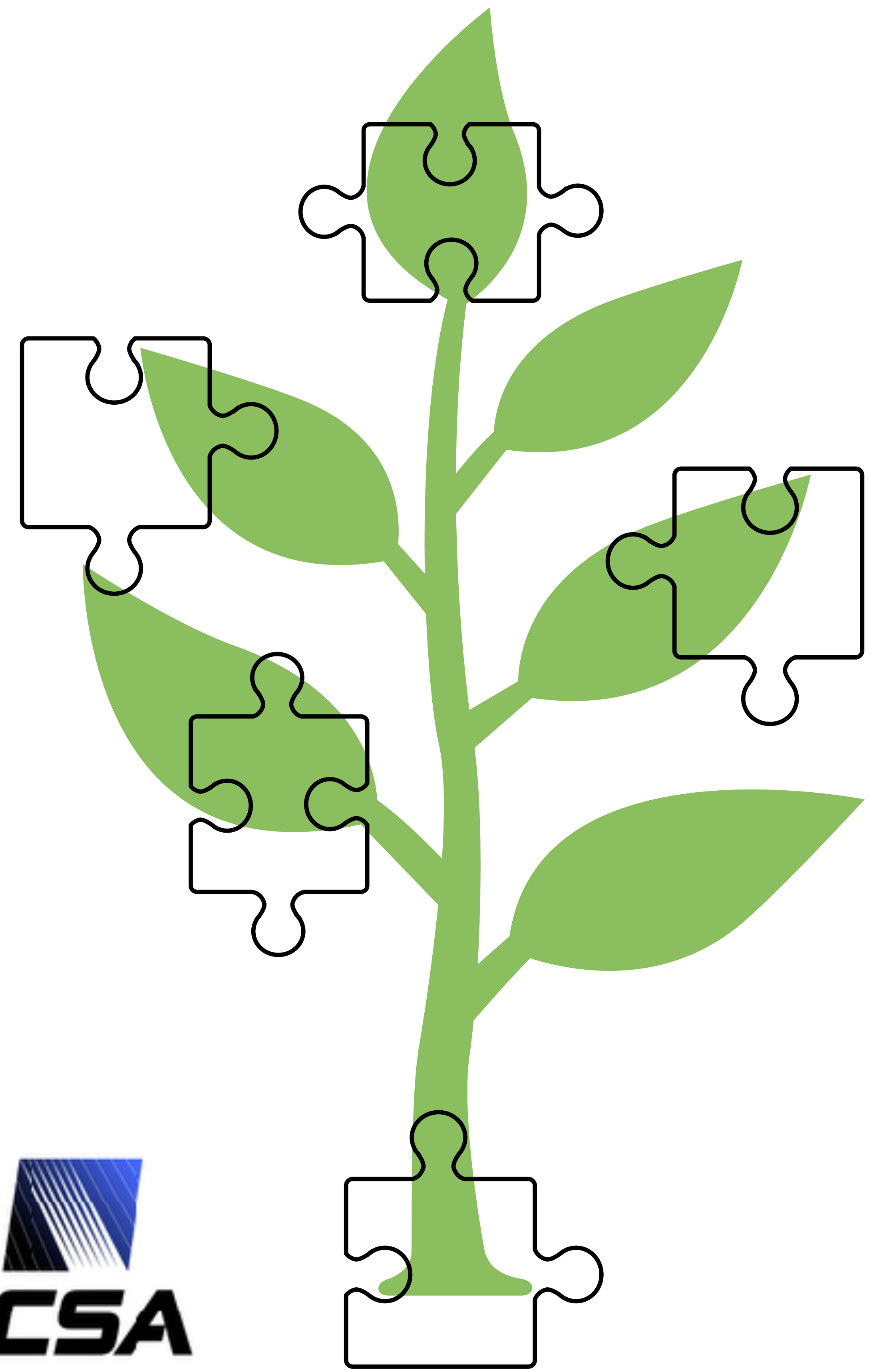


MANY EXISTING CROP BIOLOGY MODELS



BUILD CROPS IN SILICO

Combine models to create simulations that provide new insights into plant biology.



Crops in silico

ILLINOIS

In partnership with
Pennsylvania State University

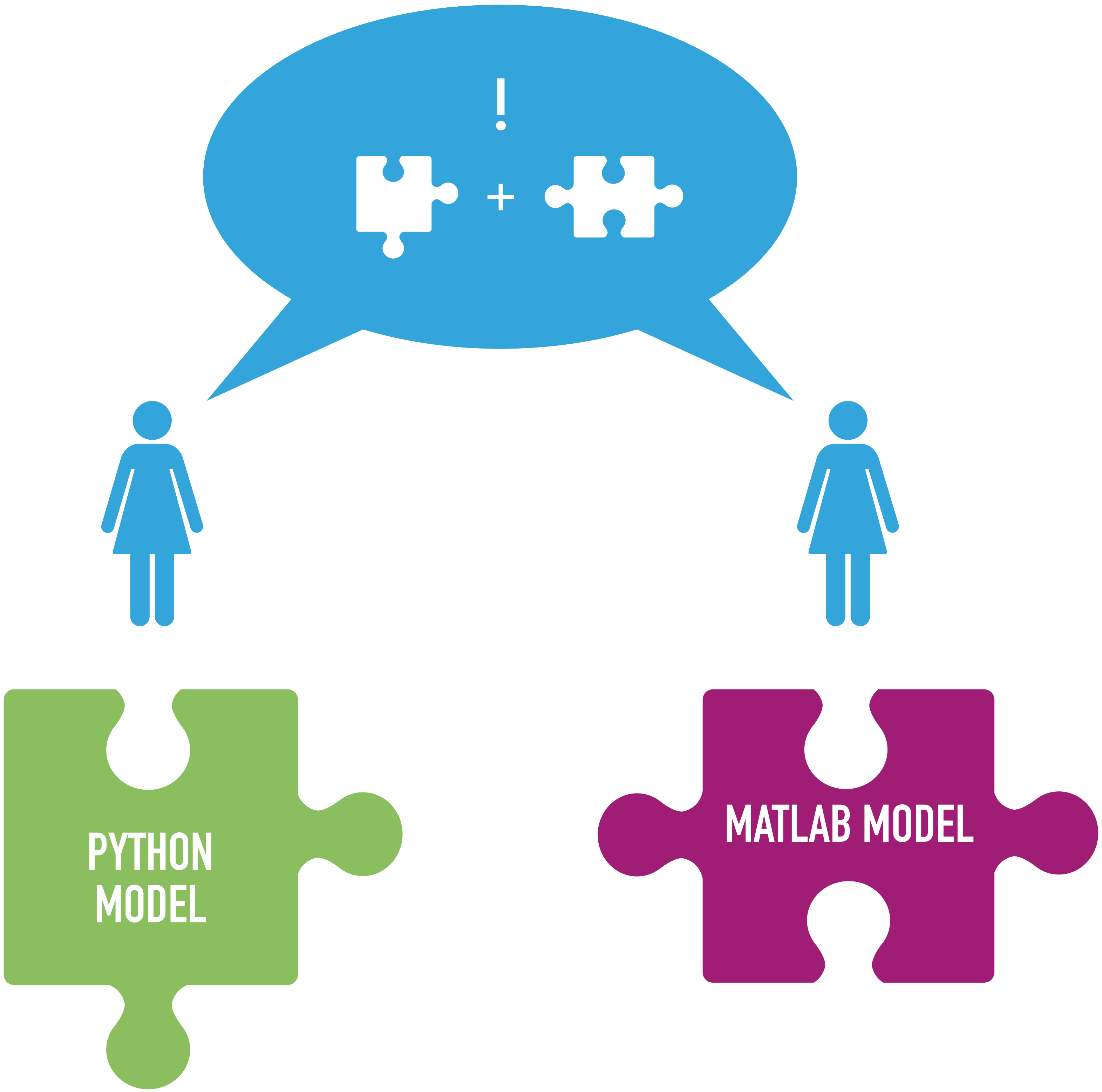
Funded by

FFAR ISEE NCSA

THE PROBLEM

MODEL INTEGRATION

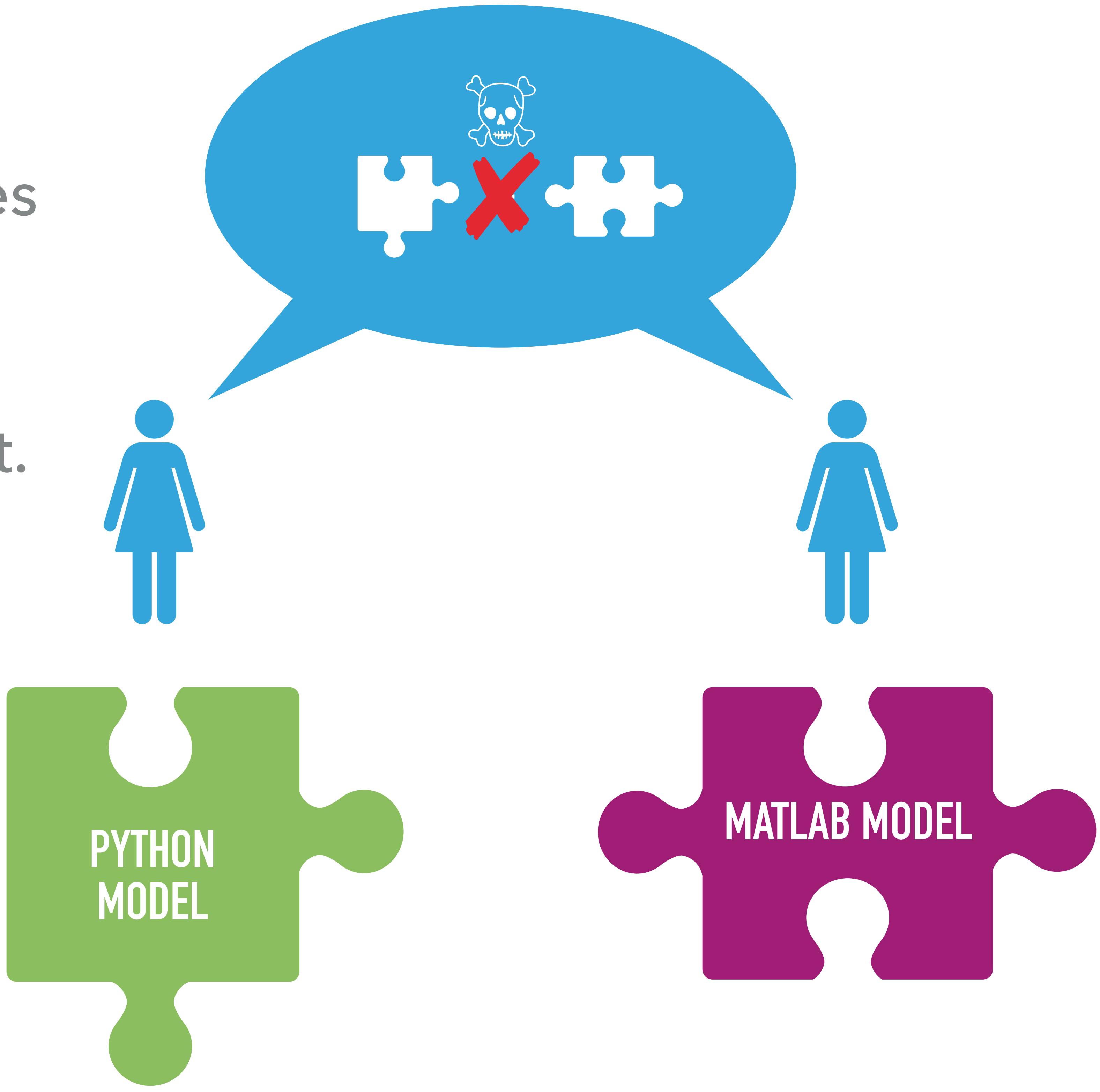
Scientists want to combine models that solve complimentary parts of a research problem.



COMMUNICATION

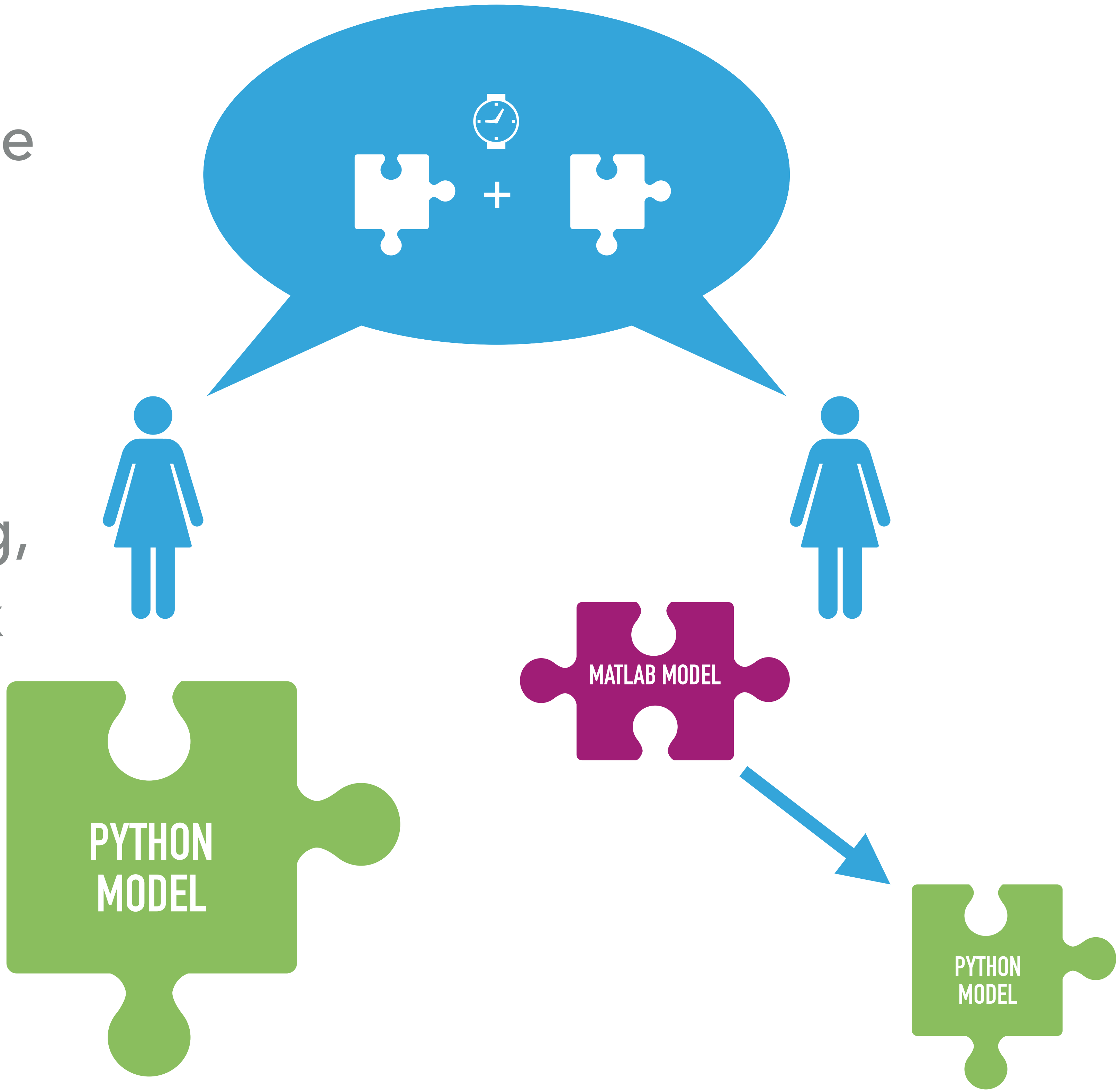
Models in different programming languages can't (always) directly communicate and are not designed to interact.

Pieces from different puzzles.



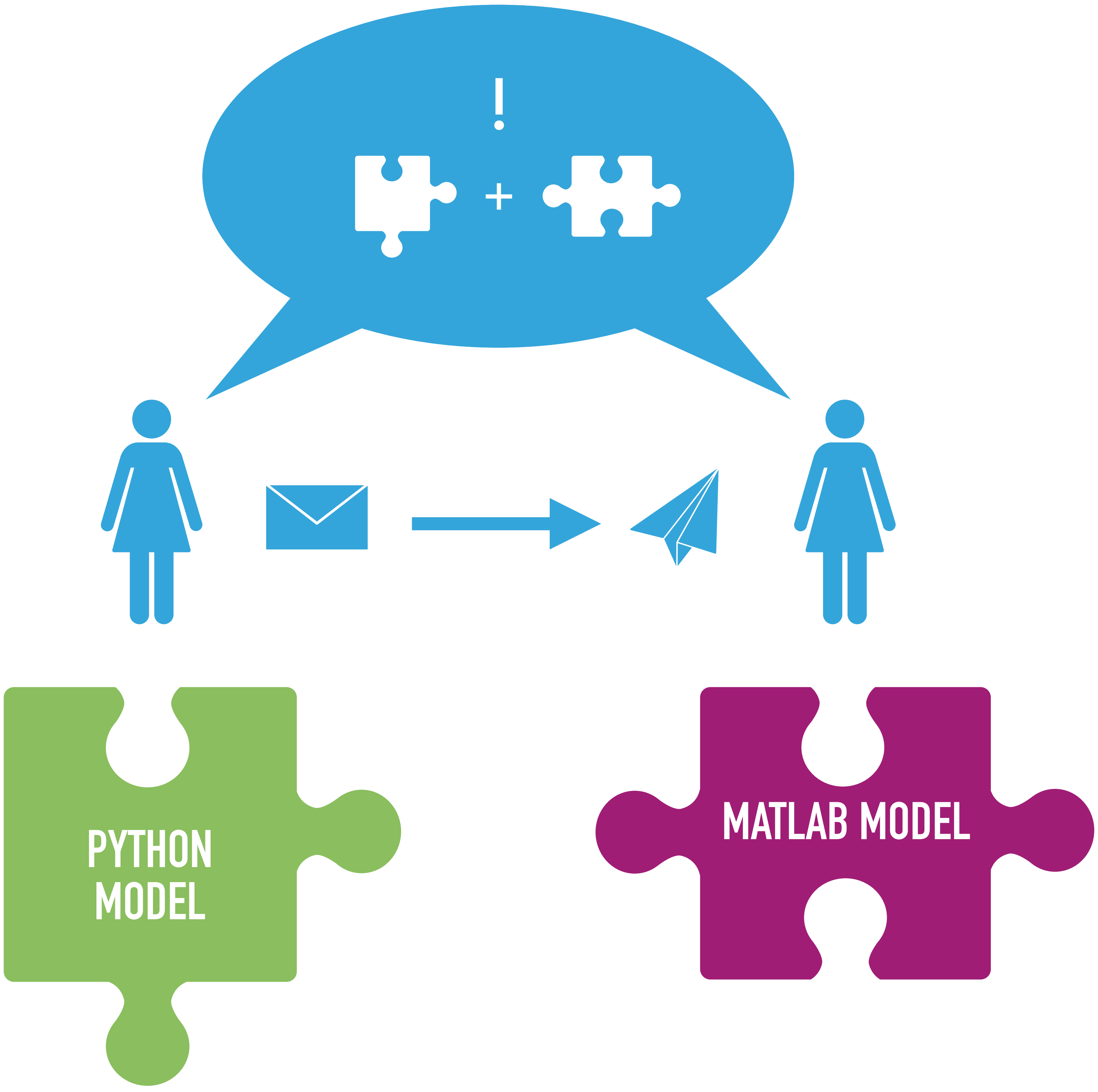
One (or both) models could be re-written to be in the same language, but...

Re-writing models can be very time consuming, particularly for complex models.



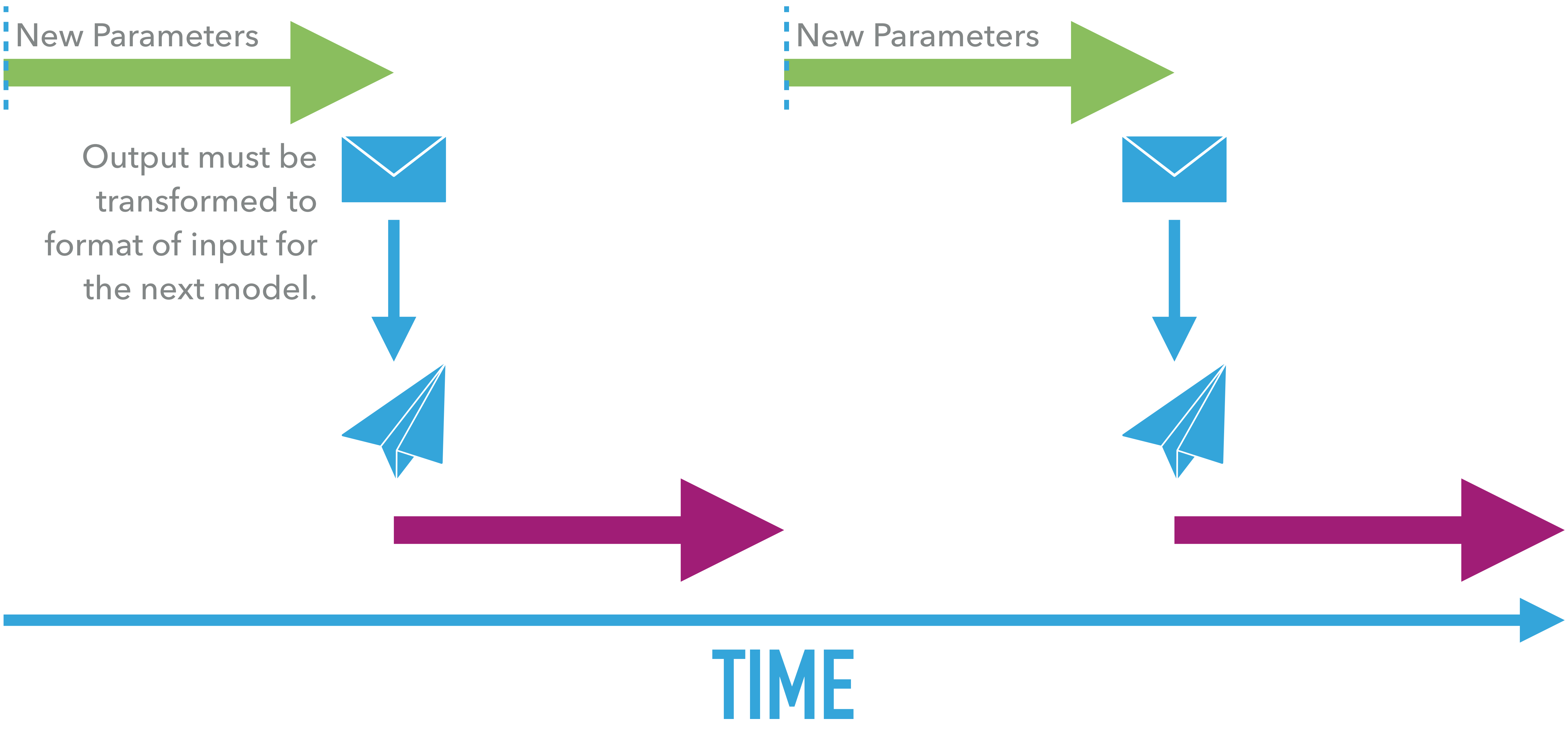
Scientists can run models independently, transforming data passed between models between runs, but…

Data transformation requires re-writing existing code or writing new code.



MODELS MUST BE RUN IN SERIAL

Communication can only be completed once the first model is done running (costly for parameter searches).



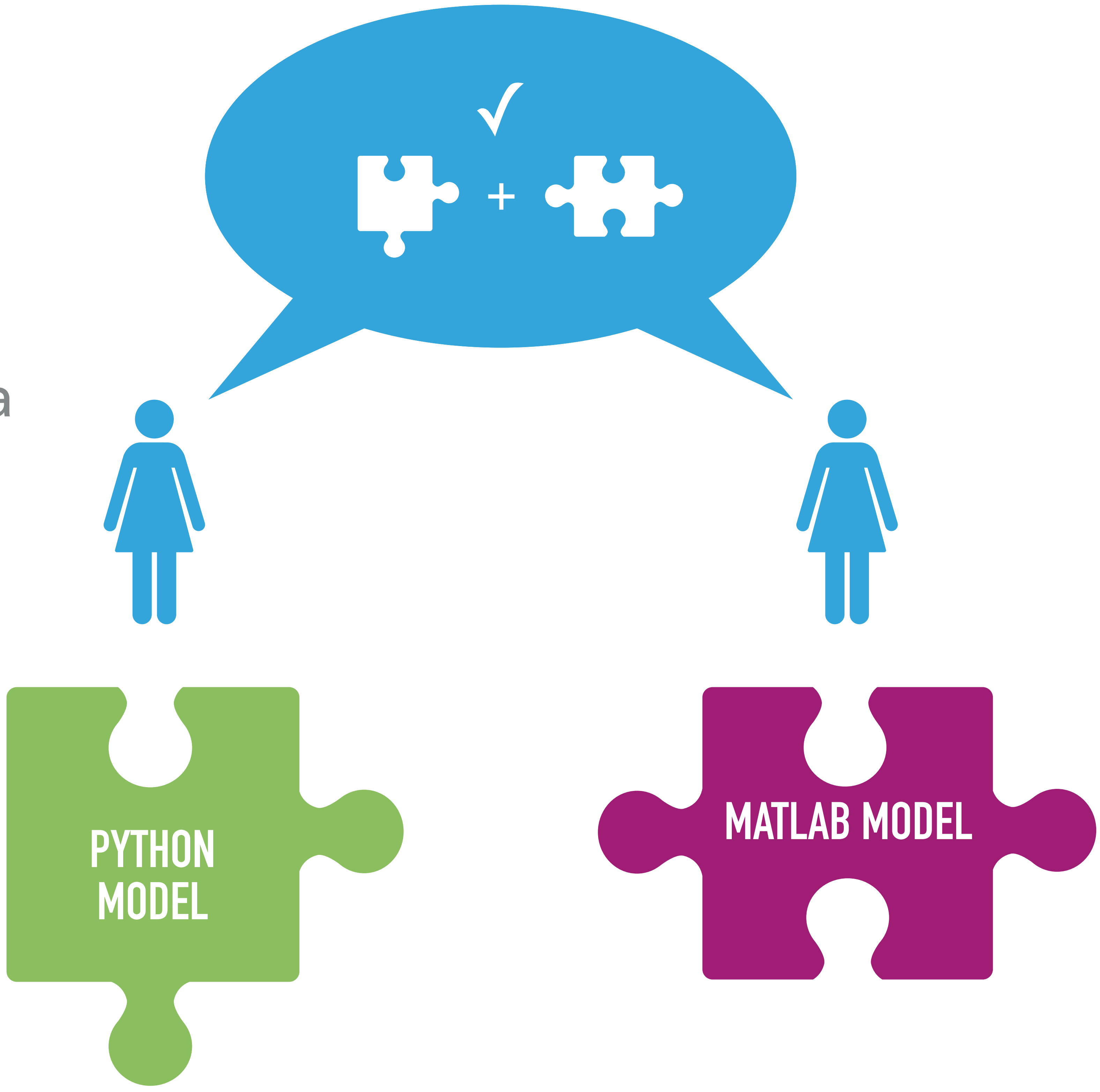
THE SOLUTION

REQUIREMENTS FOR CIS_INTERFACE

- Communication between models in different programming languages
- Parallel execution of models for improved performance
- Automated transformation of data passed between models

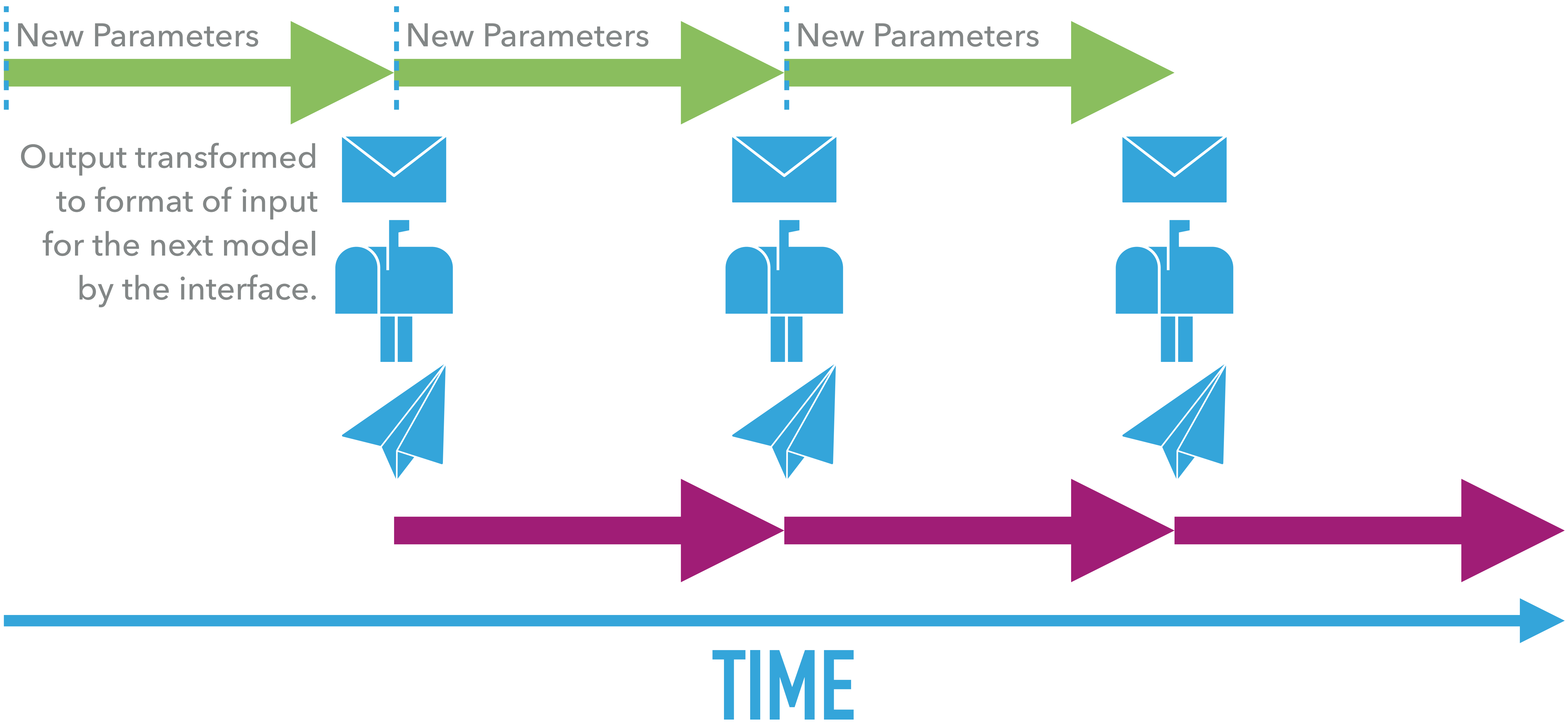
COMMUNICATION

- API calls in model's native language (Python, Matlab, C, C++) to send/recv data
- Minimal modification of source code & reusable
- No knowledge needed of communication mechanism



PARALLEL EXECUTION

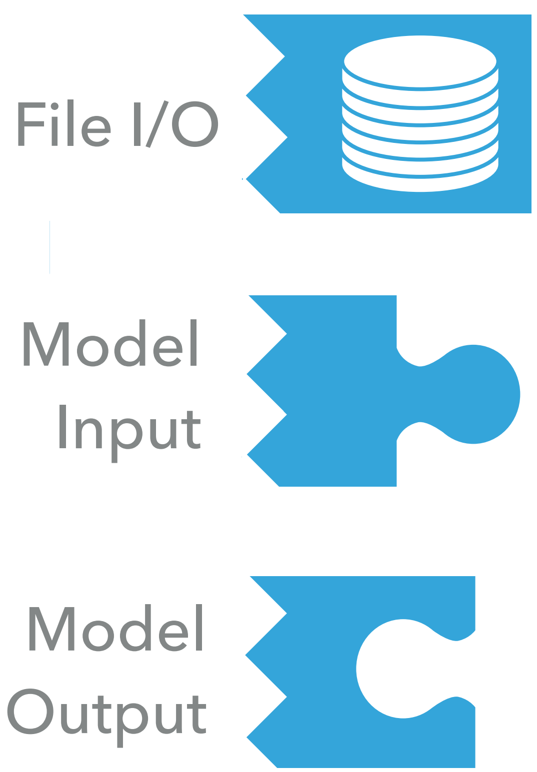
Communication is asynchronous so performance is limited only by independence of the models and the problem size.

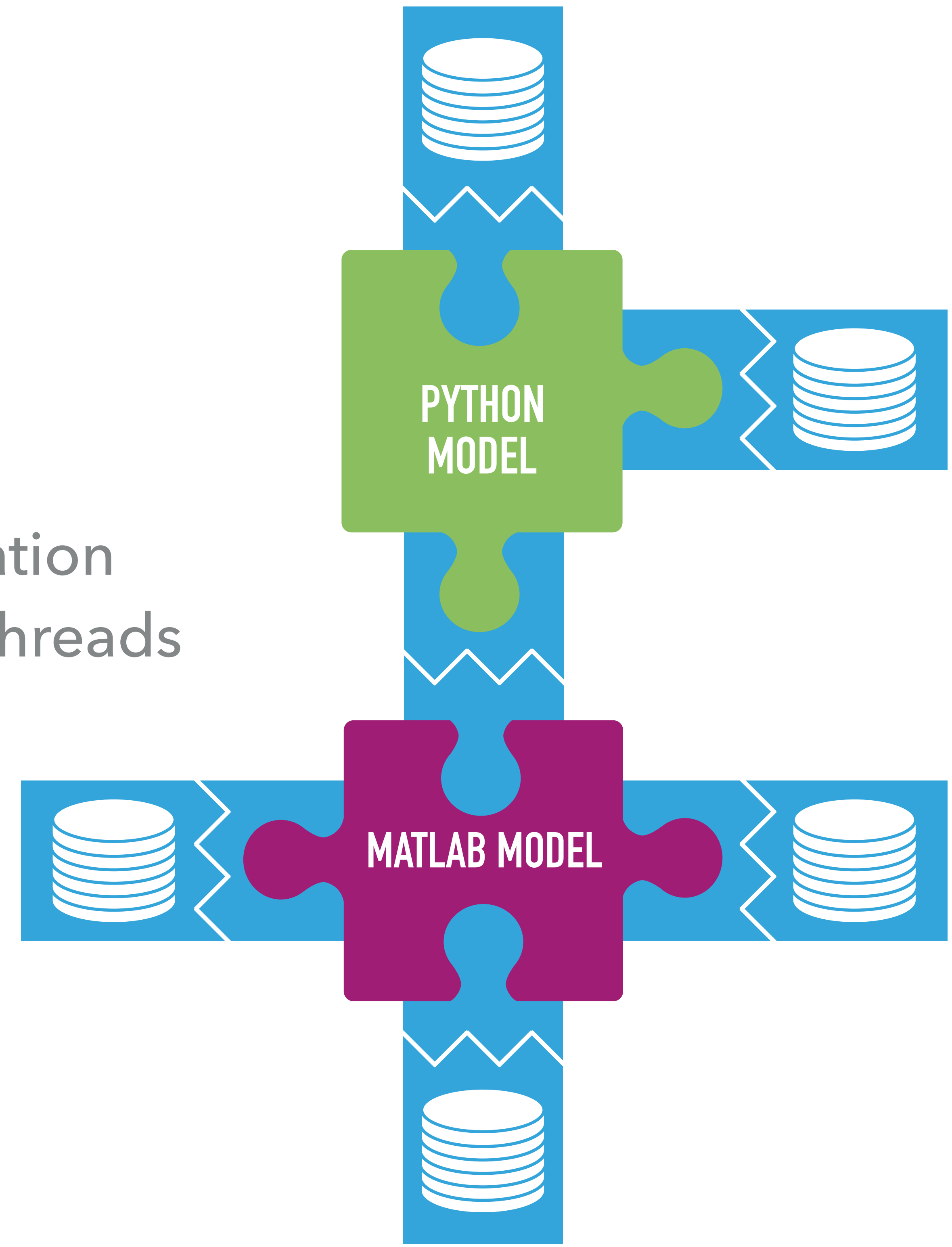


CIS_INTERFACE WORKFLOW

- Ingest information on models & connections from YAML files
- Launch models on new processes
- Coordinate asynchronous communication between models & file system using threads

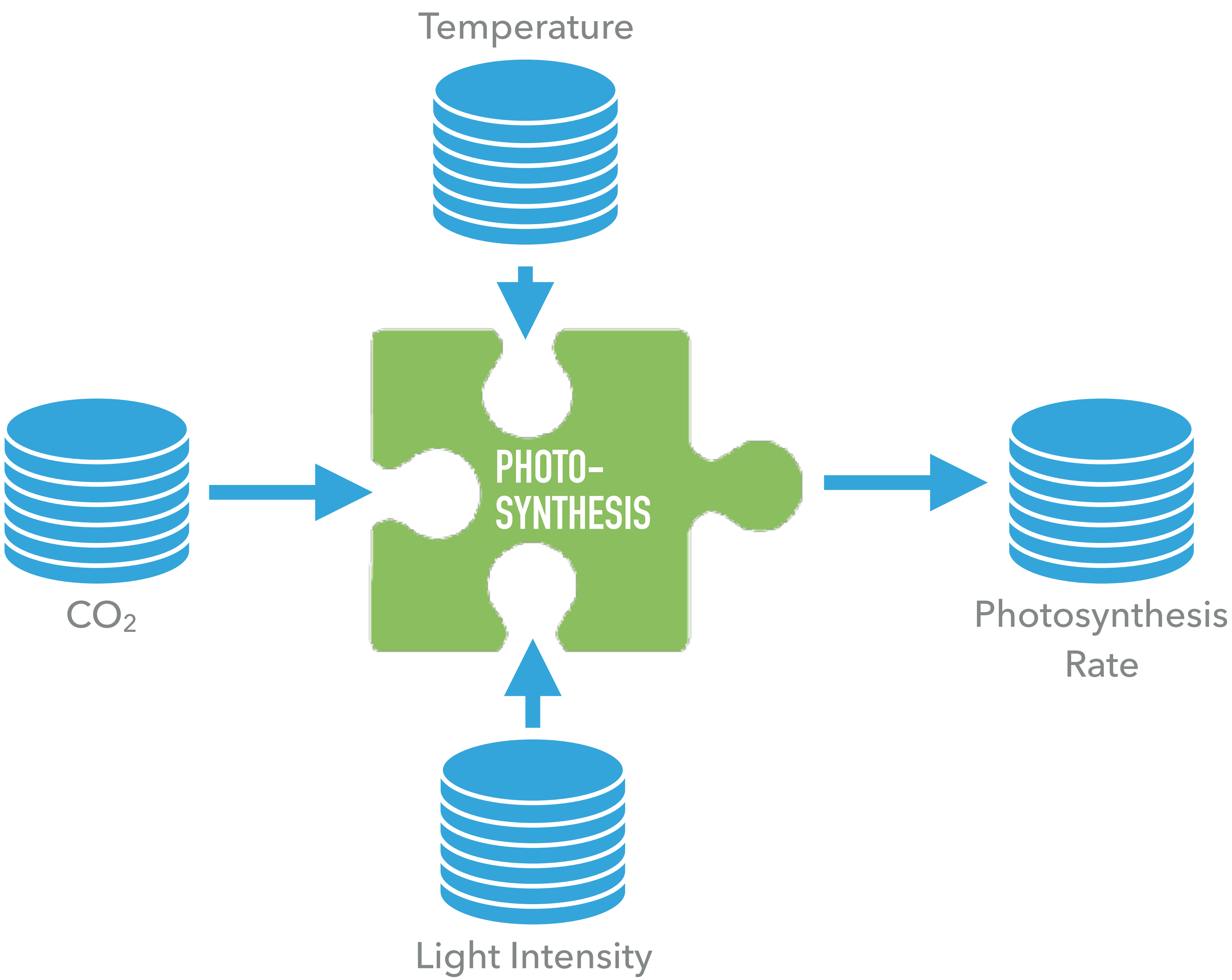
CONNECTORS





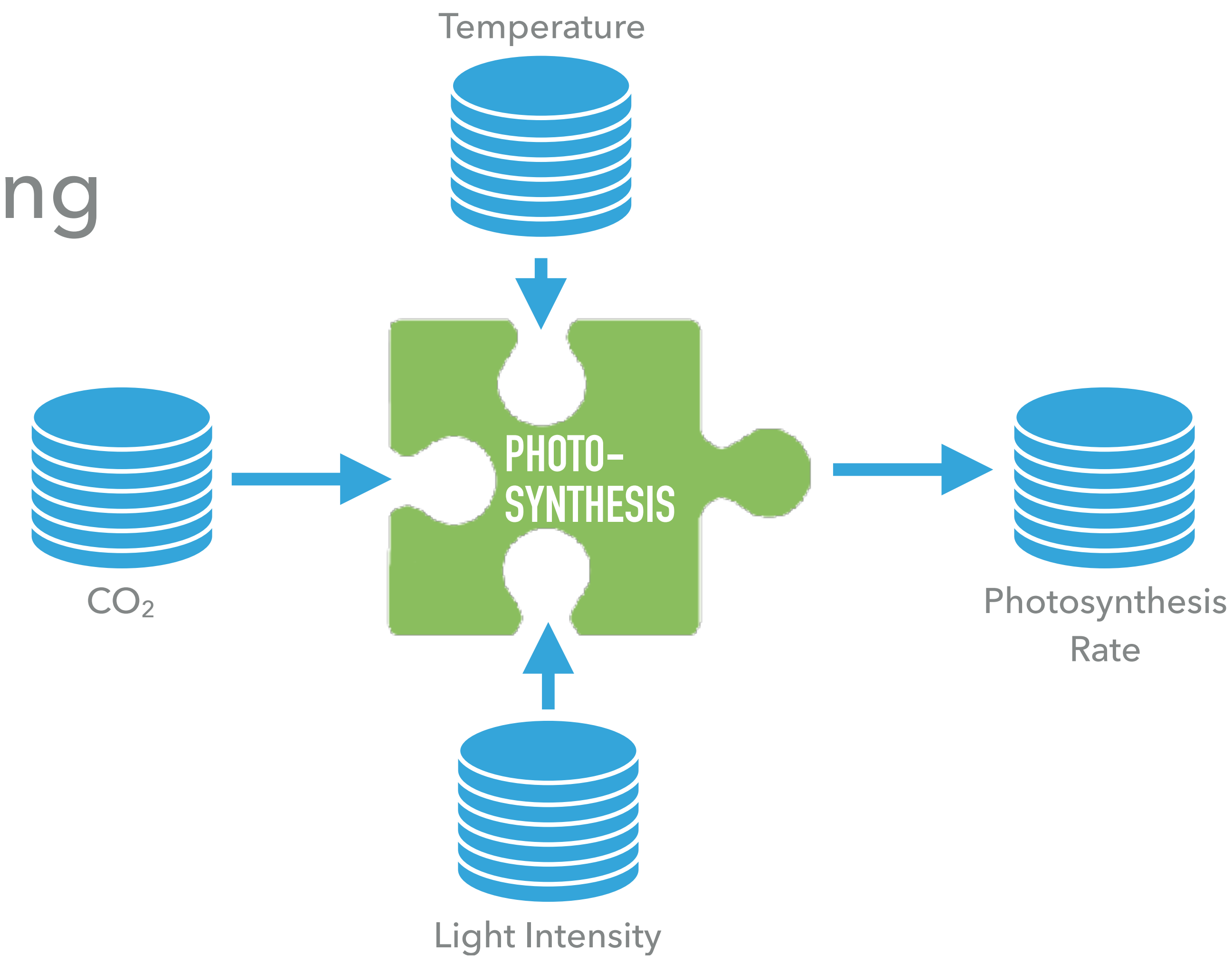
WORKED EXAMPLE

PHOTOSYNTHESIS MODEL



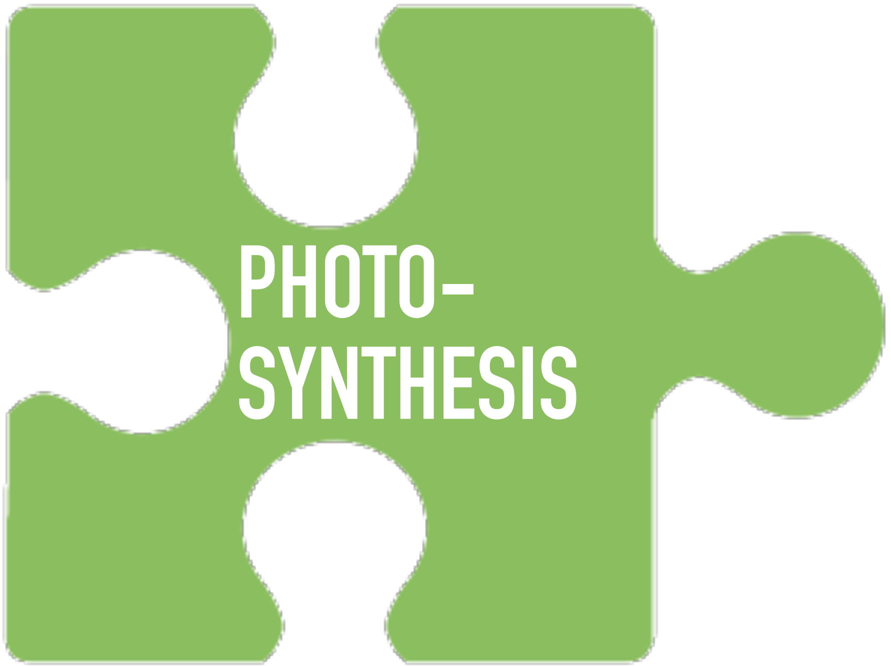
STEPS WITH CIS_INTERFACE

1. Create YAML file describing model
2. Add API calls to model
3. Create YAML file describing connections



MODEL YAML

- Language
- Source location
- Input/Output channels
- Re-usable

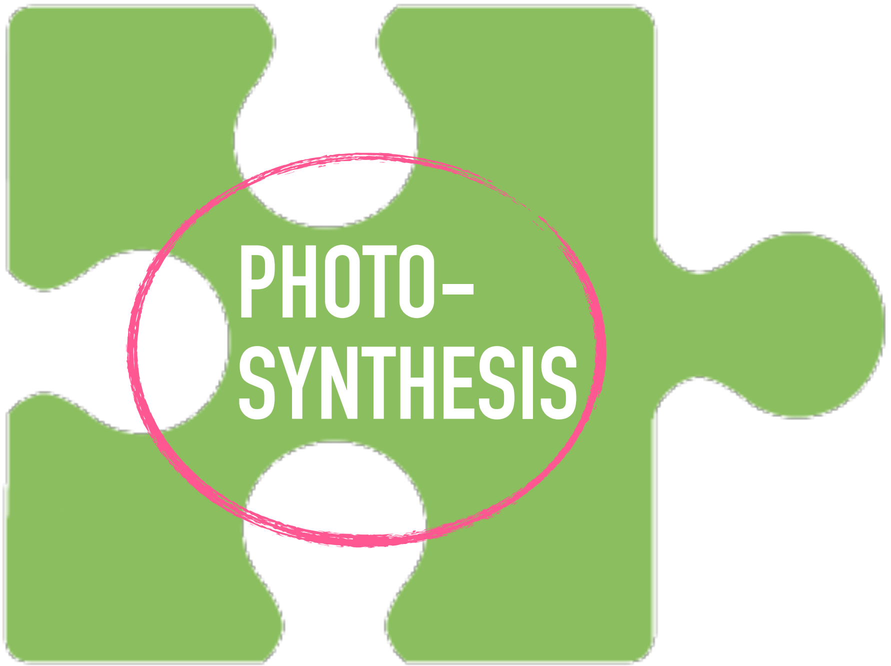


photosynthesis.yml

```
model:
```

MODEL YAML

- Language
- Source location
- Input/Output channels
- Re-usable

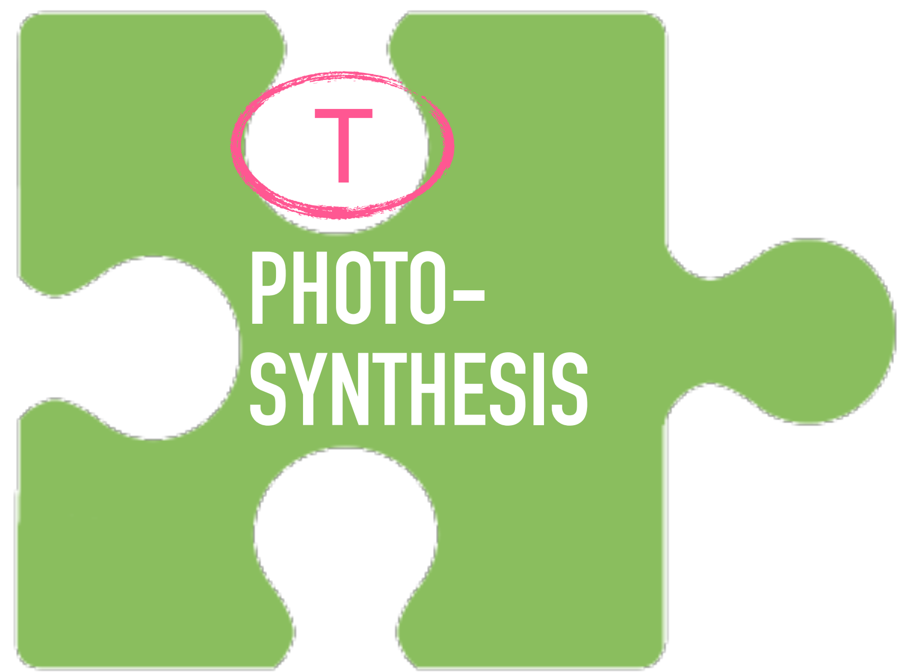


photosynthesis.yml

```
model:
  name: PhotosynthesisModel
  language: python
  args: ./photosynthesis.py
```

MODEL YAML

- Language
- Source location
- Input/Output channels
- Re-usable

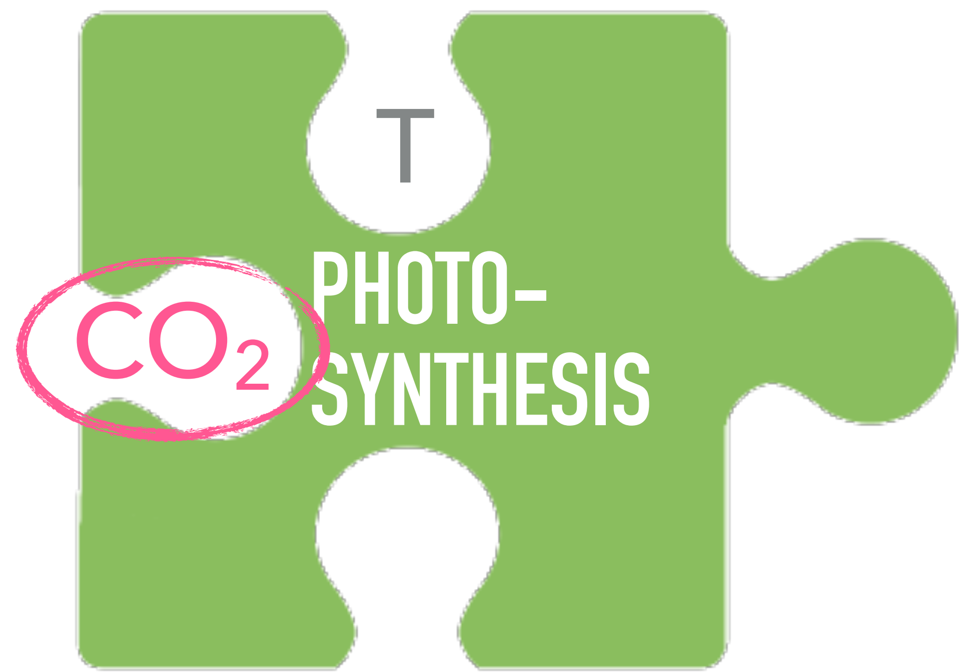


photosynthesis.yml

```
model:
  name: PhotosynthesisModel
  language: python
  args: ./photosynthesis.py
  inputs:
    - temperature
```

MODEL YAML

- Language
- Source location
- Input/Output channels
- Re-usable

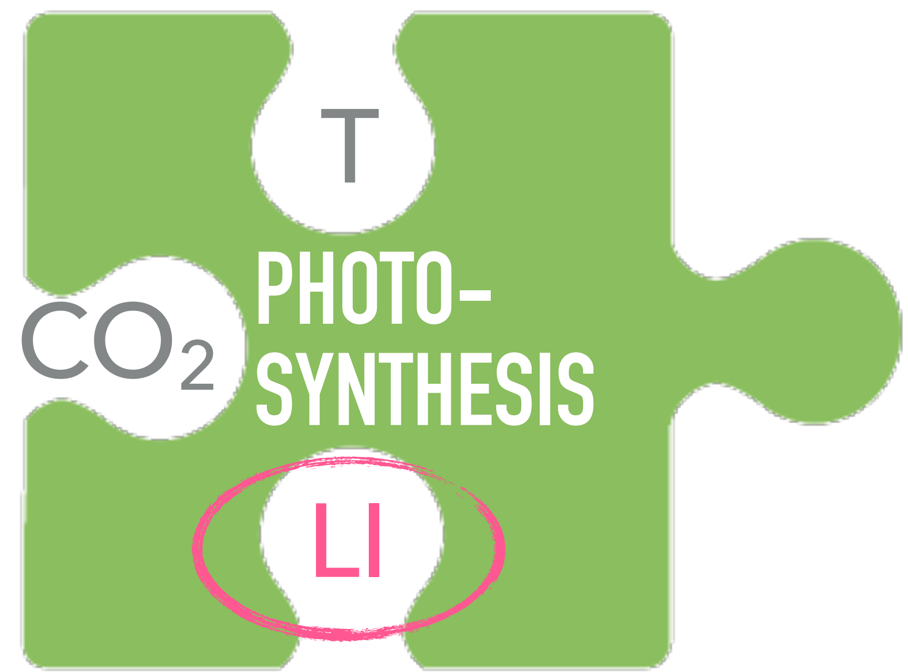


photosynthesis.yml

```
model:
  name: PhotosynthesisModel
  language: python
  args: ./photosynthesis.py
  inputs:
    - temperature
    - co2



```

MODEL YAML

- Language
- Source location
- Input/Output channels
- Re-usable

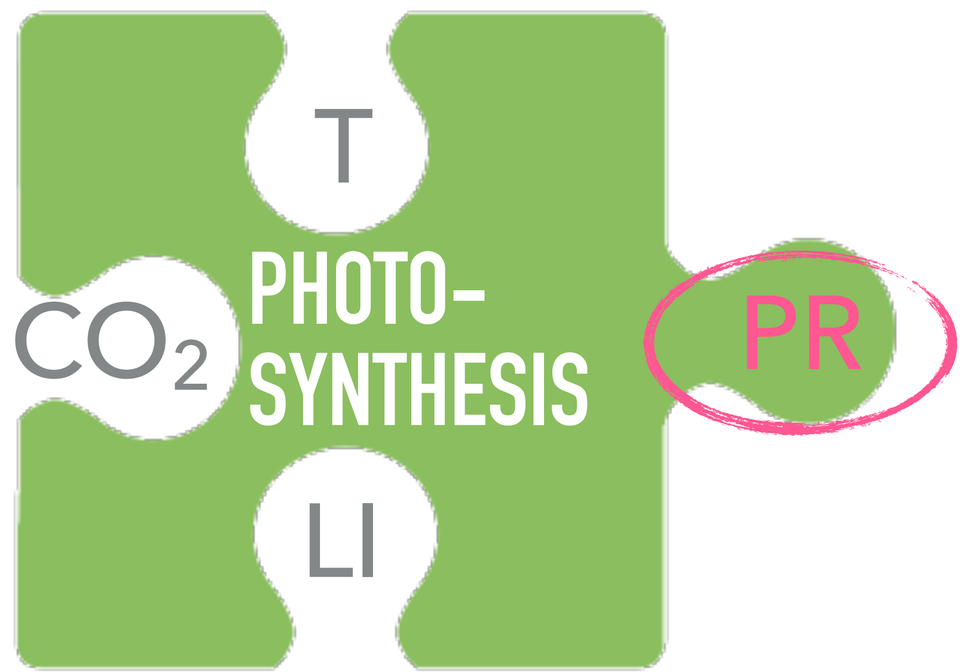


photosynthesis.yml

```
model:
  name: PhotosynthesisModel
  language: python
  args: ./photosynthesis.py
  inputs:
    - temperature
    - co2
    - light_intensity


```

MODEL YAML

- Language
- Source location
- Input/Output channels
- Re-usable



photosynthesis.yml

```yaml
model:
  name: PhotosynthesisModel
  language: python
  args: ./photosynthesis.py
  inputs:
    - temperature
    - co2
    - light_intensity
  outputs:
    - photosynthesis_rate
```

MODEL TRANSFORMATION

photosynthesis.py

Original model code.

```
#!/usr/bin/python
import sys
import numpy as np

def calc_photo_rate(T, CO2, light):
    return light * CO2 / T

if __name__ == ‘__main__’:
    # File names passed as command line arguments
    input_filename_T = sys.argv[1]
    input_filename_CO2 = sys.argv[2]
    input_filename_light = sys.argv[3]
    output_filename_photo = sys.argv[4]

    # Load input data from files
    T_in = np.loadtxt(input_filename_T)
    CO2_in = np.loadtxt(input_filename_CO2)
    light_in = np.loadtxt(input_filename_light)

    # Calculate photosynthesis
    photo_out = calc_photo_rate(T_in, CO2_in, light_in)

    # Save output to file
    np.savetxt(output_filename_photo, photo_out)
```

MODEL TRANSFORMATION

photosynthesis.py

```
#!/usr/bin/python
from cis_interface.interface import CisInput, CisOutput


def calc_photo_rate(T, CO2, light):
    return light * CO2 / T

if __name__ == ‘__main__’:
    # File names passed as command line arguments
    input_filename_T = sys.argv[1]
    input_filename_CO2 = sys.argv[2]
    input_filename_light = sys.argv[3]
    output_filename_photo = sys.argv[4]

    # Load input data from files
    T_in = np.loadtxt(input_filename_T)
    CO2_in = np.loadtxt(input_filename_CO2)
    light_in = np.loadtxt(input_filename_light)

    # Calculate photosynthesis
    photo_out = calc_photo_rate(T_in, CO2_in, light_in)

    # Save output to file
    np.savetxt(output_filename_photo, photo_out)
```

Import cis_interface API

MODEL TRANSFORMATION

photosynthesis.py

```
#!/usr/bin/python
from cis_interface.interface import CisInput, CisOutput


def calc_photo_rate(T, CO2, light):
    return light * CO2 / T

if __name__ == ‘__main__’:
    # Input channels from names
    input_channel_T = CisInput('temperature')
    input_channel_CO2 = CisInput('co2')
    input_channel_light = CisInput('light_intensity')
    output_channel_photo = CisOutput('photosynthesis_rate', '%lf\n')

    # Load input data from files
    T_in = np.loadtxt(input_filename_T)
    CO2_in = np.loadtxt(input_filename_CO2)
    light_in = np.loadtxt(input_filename_light)

    # Calculate photosynthesis
    photo_out = calc_photo_rate(T_in, CO2_in, light_in)

    # Save output to file
    np.savetxt(output_filename_photo, photo_out)
```

Create input/output channels using channel names from the YAML.

MODEL TRANSFORMATION

photosynthesis.py

```
#!/usr/bin/python
from cis_interface.interface import CisInput, CisOutput

def calc_photo_rate(T, CO2, light):
    return light * CO2 / T

if __name__ == '__main__':
    # Input channels from names
    input_channel_T = CisInput('temperature')
    input_channel_CO2 = CisInput('co2')
    input_channel_light = CisInput('light_intensity')
    output_channel_photo = CisOutput('photosynthesis_rate', '%lf\n')

    # Receive input data from channels
    flag_T, T_in = input_channel_T.recv()
    flag_CO2, CO2_in = input_channel_CO2.recv()
    flag_light, light_in = input_channel_light.recv()

    # Calculate photosynthesis
    photo_out = calc_photo_rate(T_in, CO2_in, light_in)

    # Save output to file
    np.savetxt(output_filename_photo, photo_out)
```

Receive input from channels

MODEL TRANSFORMATION

photosynthesis.py

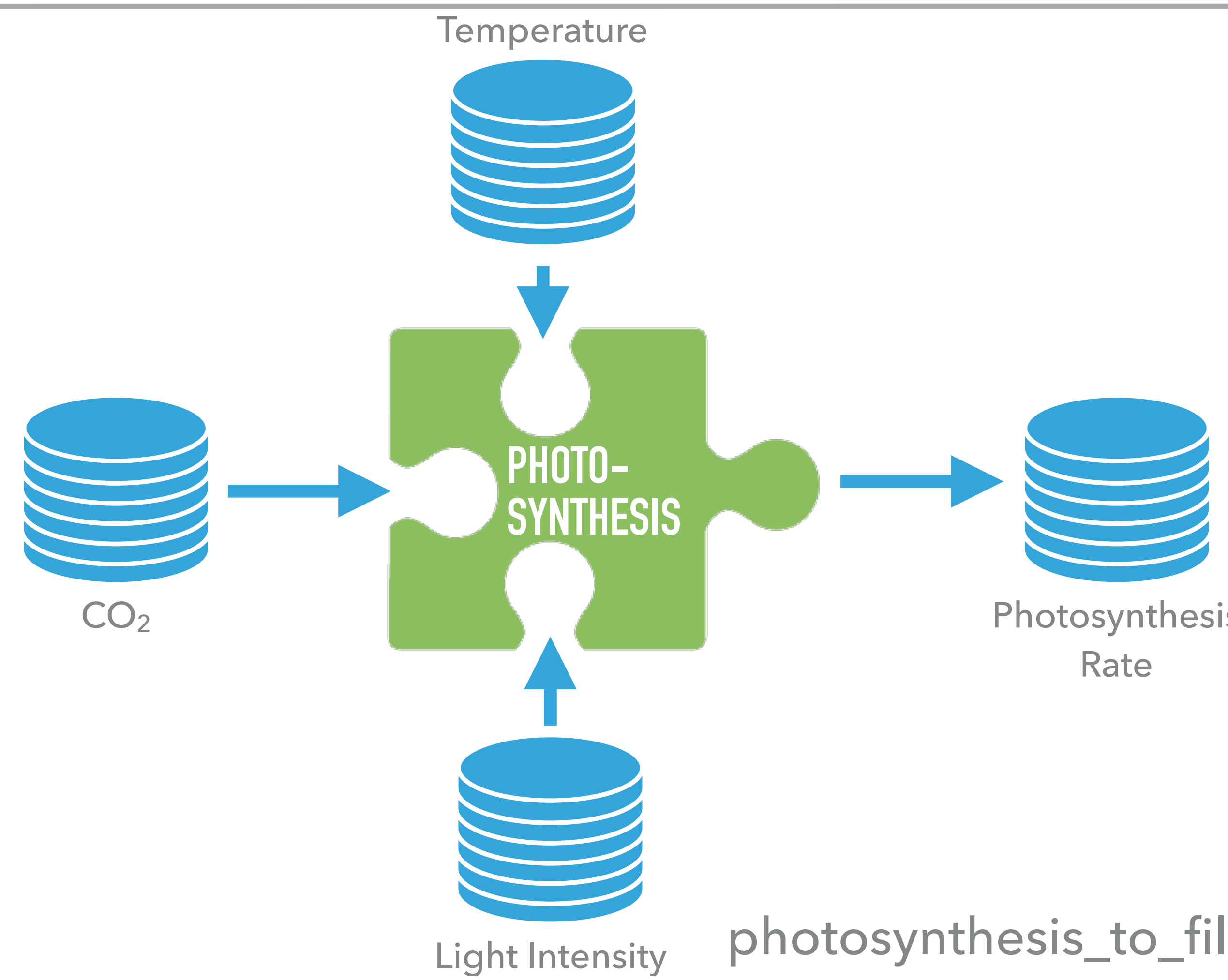
```
#!/usr/bin/python
from cis_interface.interface import CisInput, CisOutput


def calc_photo_rate(T, CO2, light):
    return light * CO2 / T

if __name__ == ‘__main__’:
    # Input channels from names
    input_channel_T = CisInput('temperature')
    input_channel_CO2 = CisInput('co2')
    input_channel_light = CisInput('light_intensity')
    output_channel_photo = CisOutput('photosynthesis_rate', '%lf\n')

    # Receive input data from channels
    flag_T, T_in = input_channel_T.recv()
    flag_CO2, CO2_in = input_channel_CO2.recv()
    flag_light, light_in = input_channel_light.recv()

    # Calculate photosynthesis
    photo_out = calc_photo_rate(T_in, CO2_in, light_in)

    # Send output to channel
    flag = output_channel_photo.send(photo_out)
```

Send output to channel

CONNECTION YAML

- Pairs of connections between model input/outputs and other models or files

- One per network



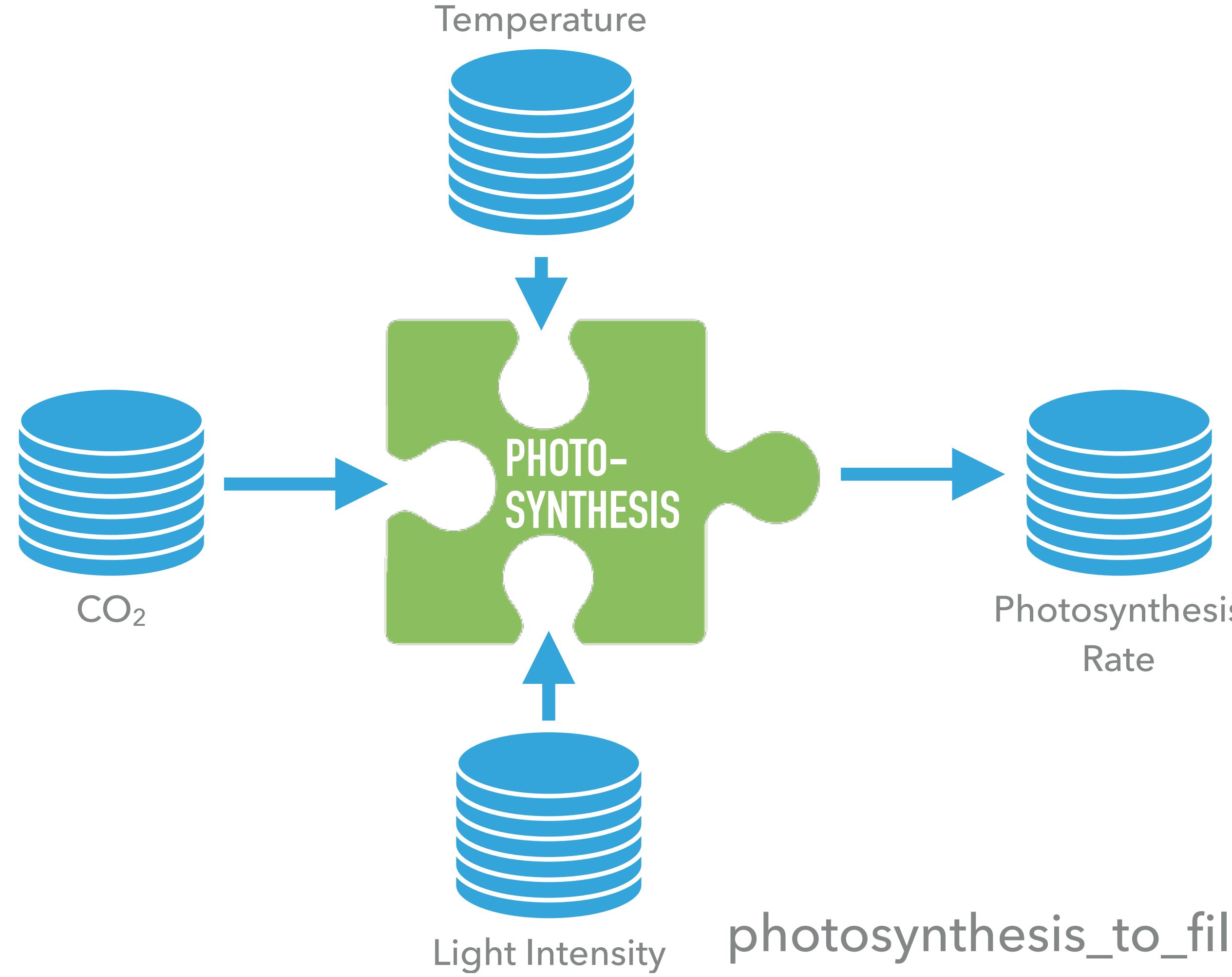
photosynthesis_to_file.yml

```
connections:
```

WORKED EXAMPLE: CONNECTION TO FILE

CONNECTION YAML

- Pairs of connections between model input/outputs and other models or files
- One per network

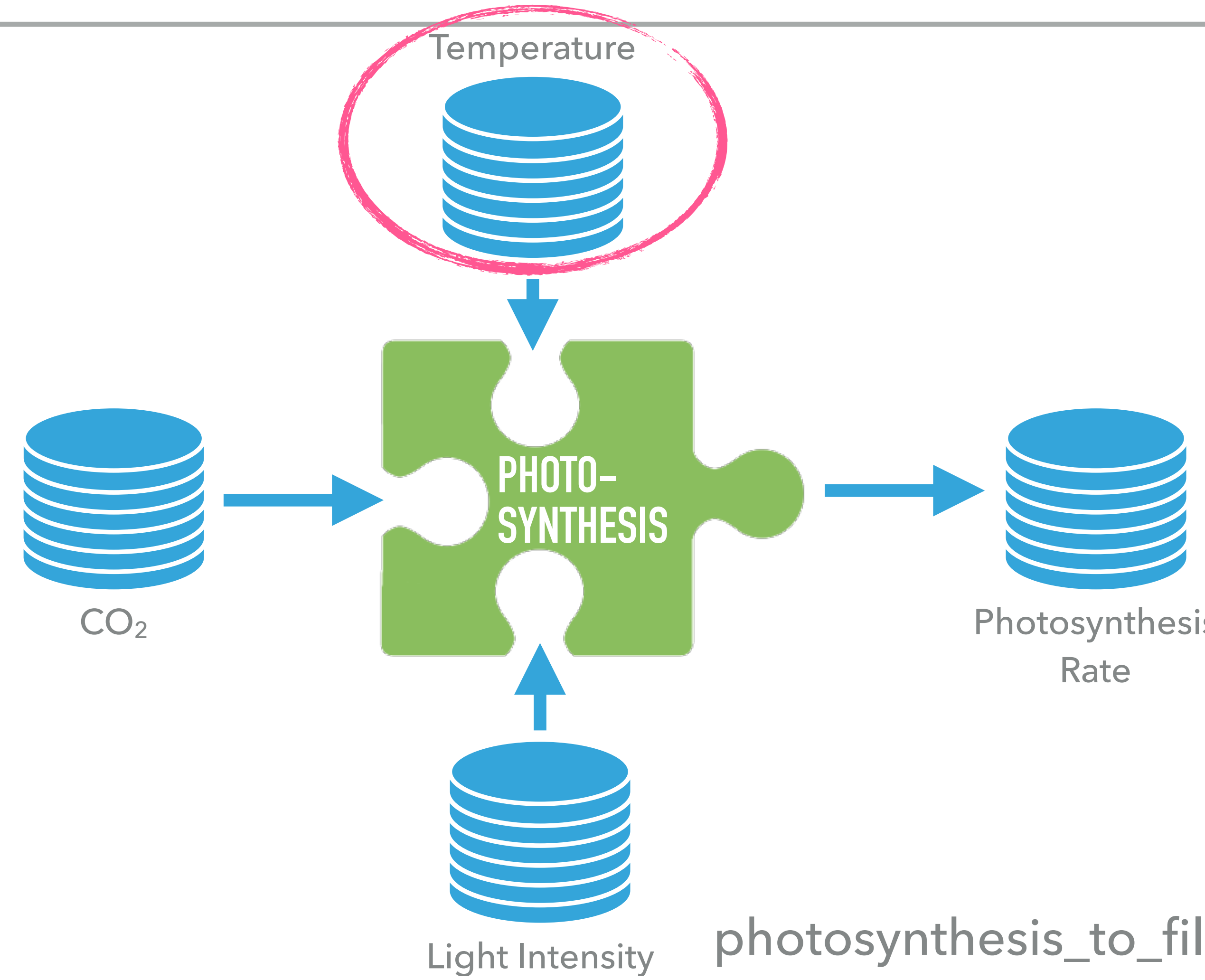


photosynthesis_to_file.yml

```yaml
connections:
  - input: ./Input/temperature.txt
    output: temperature
    filetype: table
```

CONNECTION YAML

- Pairs of connections between model input/outputs and other models or files
- One per network

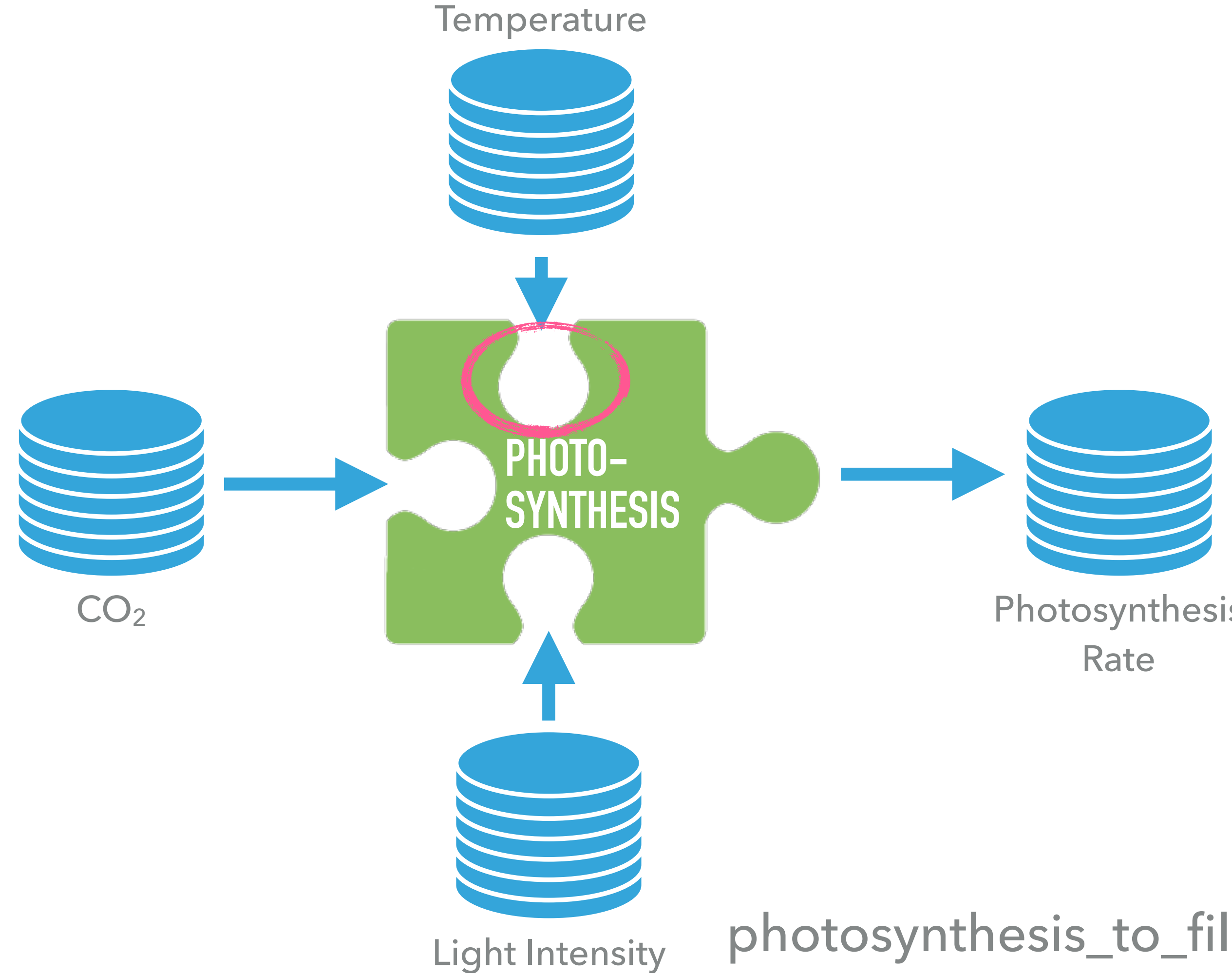


photosynthesis_to_file.yml

```yaml
connections:
  - input: ./Input/temperature.txt
    output: temperature
    filetype: table
```

CONNECTION YAML

- Pairs of connections between model input/outputs and other models or files
- One per network

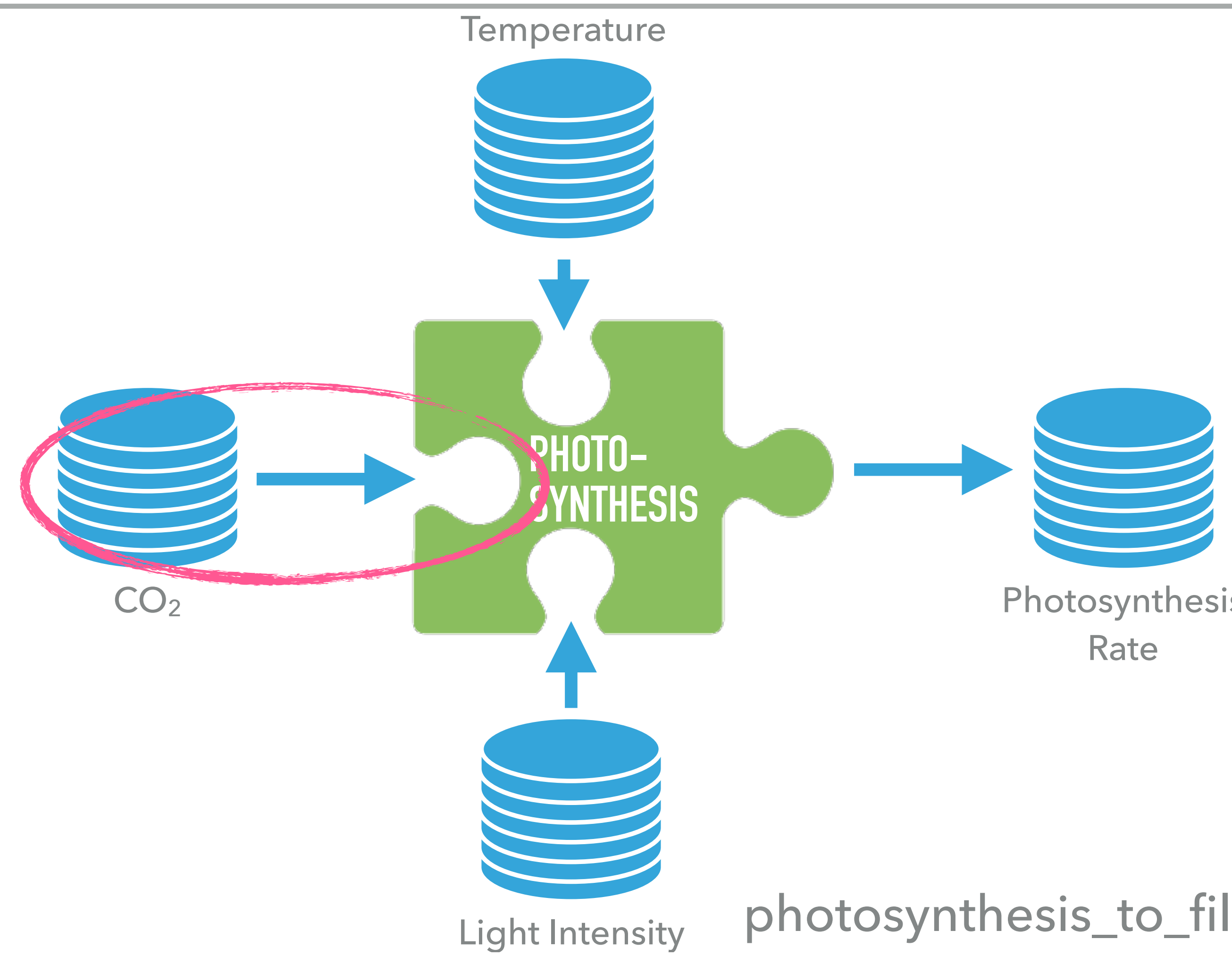


photosynthesis_to_file.yml

```
connections:
  - input: ./Input/temperature.txt
    output: temperature
    filetype: table
```

CONNECTION YAML

- Pairs of connections between model input/outputs and other models or files
- One per network



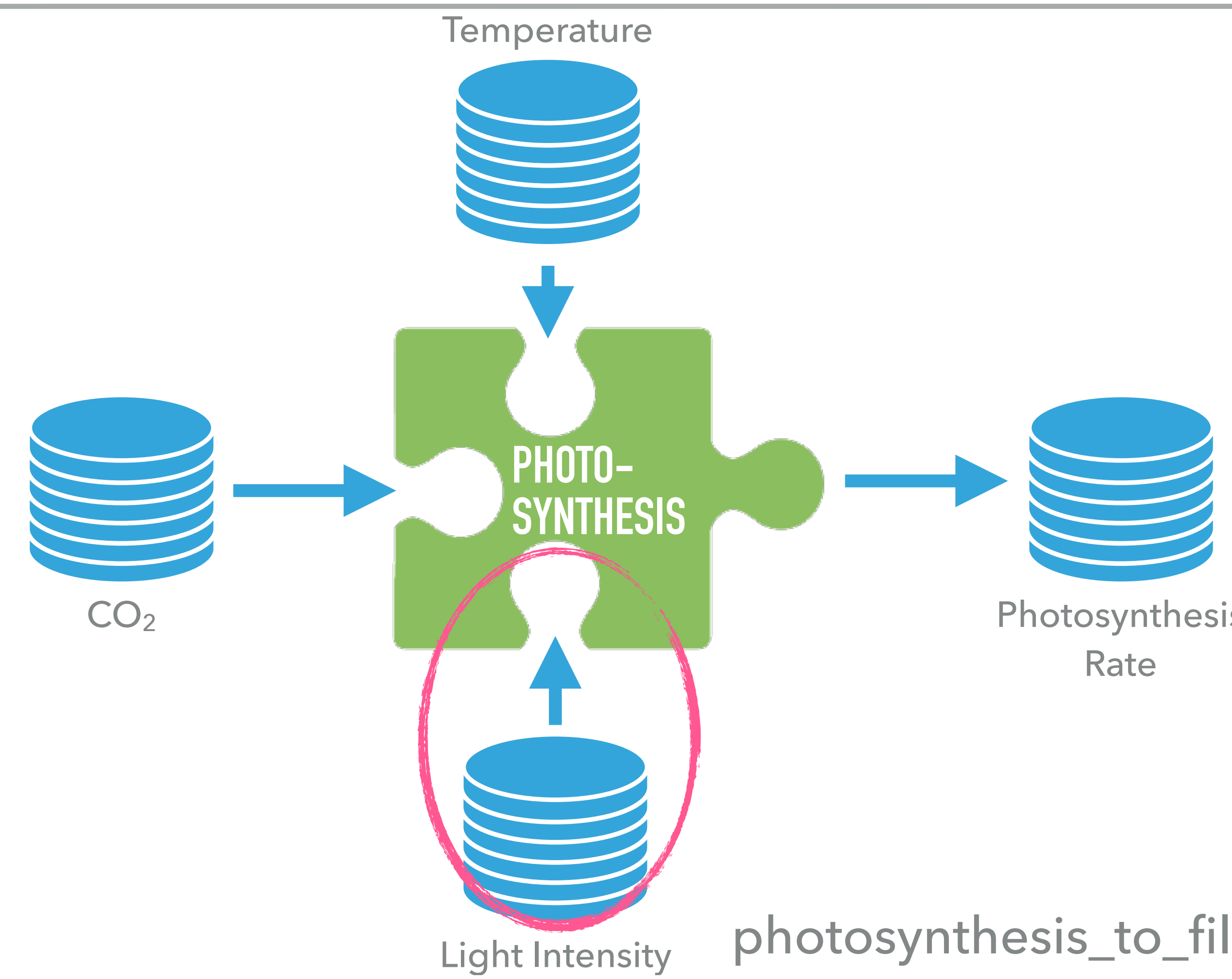
photosynthesis_to_file.yml

```yaml
connections:
  - input: ./Input/temperature.txt
    output: temperature
    filetype: table
  - input: ./Input/co2.txt
    output: co2
    filetype: table
```

CONNECTION YAML

- Pairs of connections between model input/outputs and other models or files

- One per network



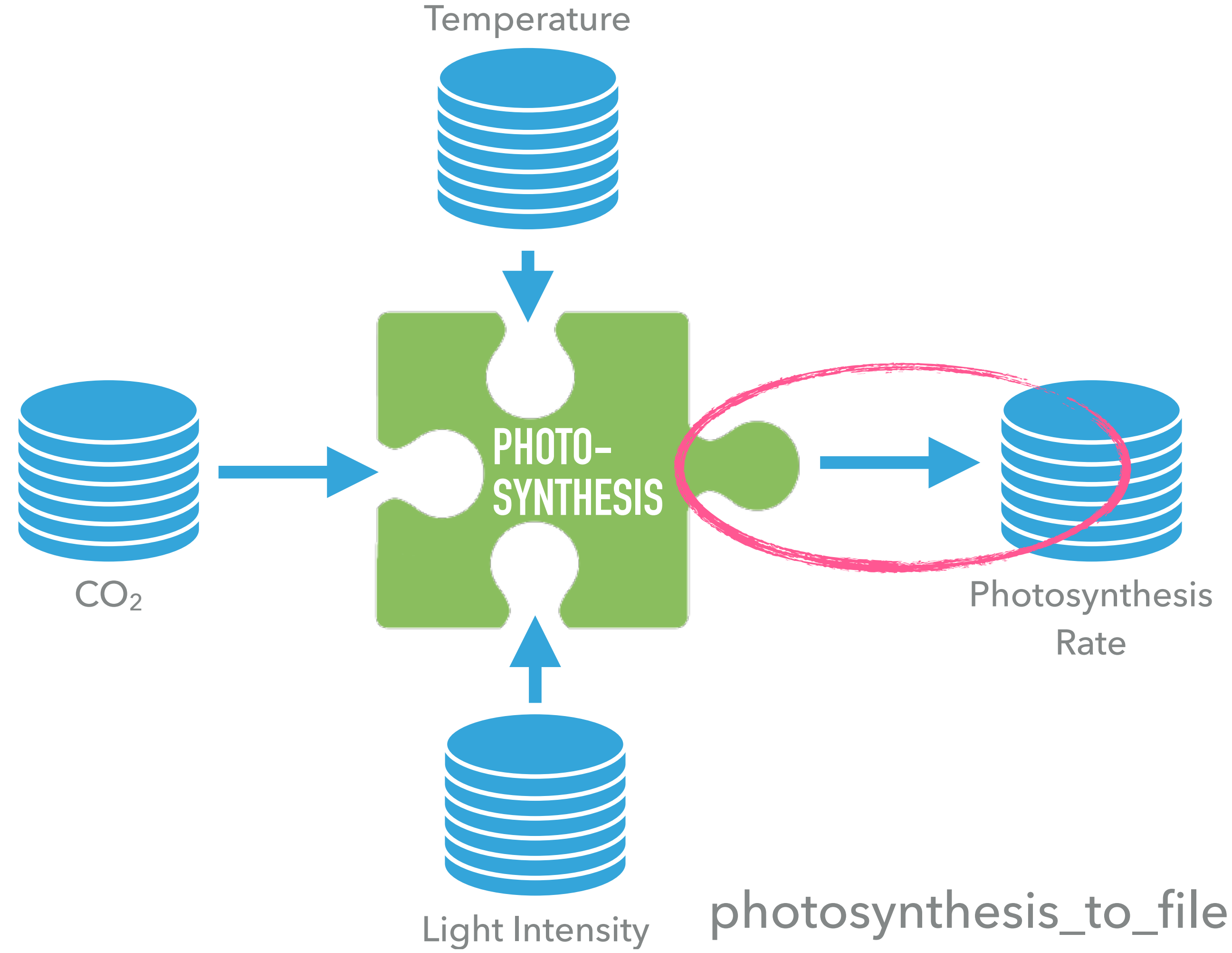
photosynthesis_to_file.yml

```
connections:
  - input: ./Input/temperature.txt
    output: temperature
    filetype: table
  - input: ./Input/co2.txt
    output: co2
    filetype: table
  - input: ./Input/light_intensity.txt
    output: light_intensity
    filetype: table
```

WORKED EXAMPLE: CONNECTION TO FILE

CONNECTION YAML

- Pairs of connections between model input/outputs and other models or files
- One per network

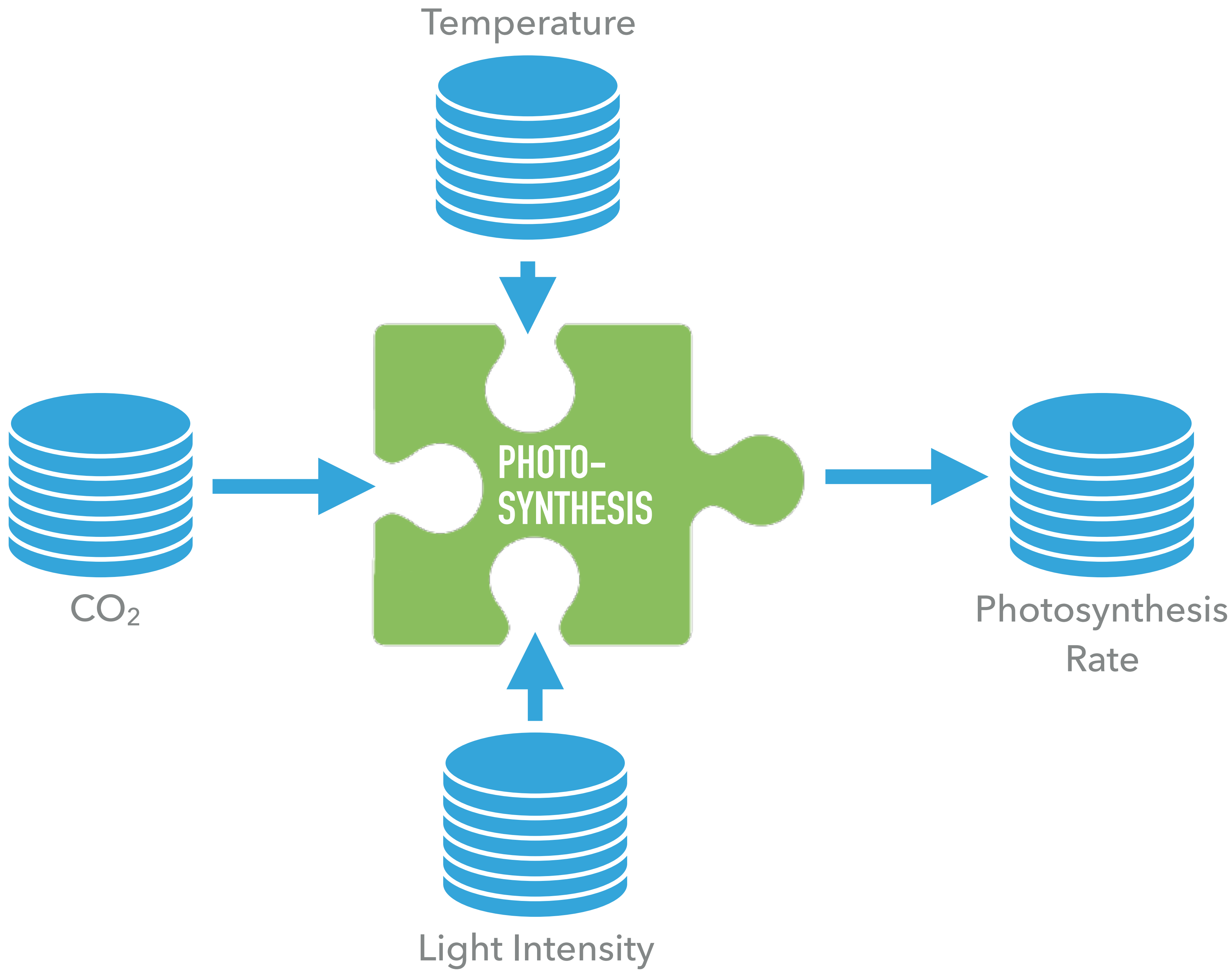


photosynthesis_to_file.yml

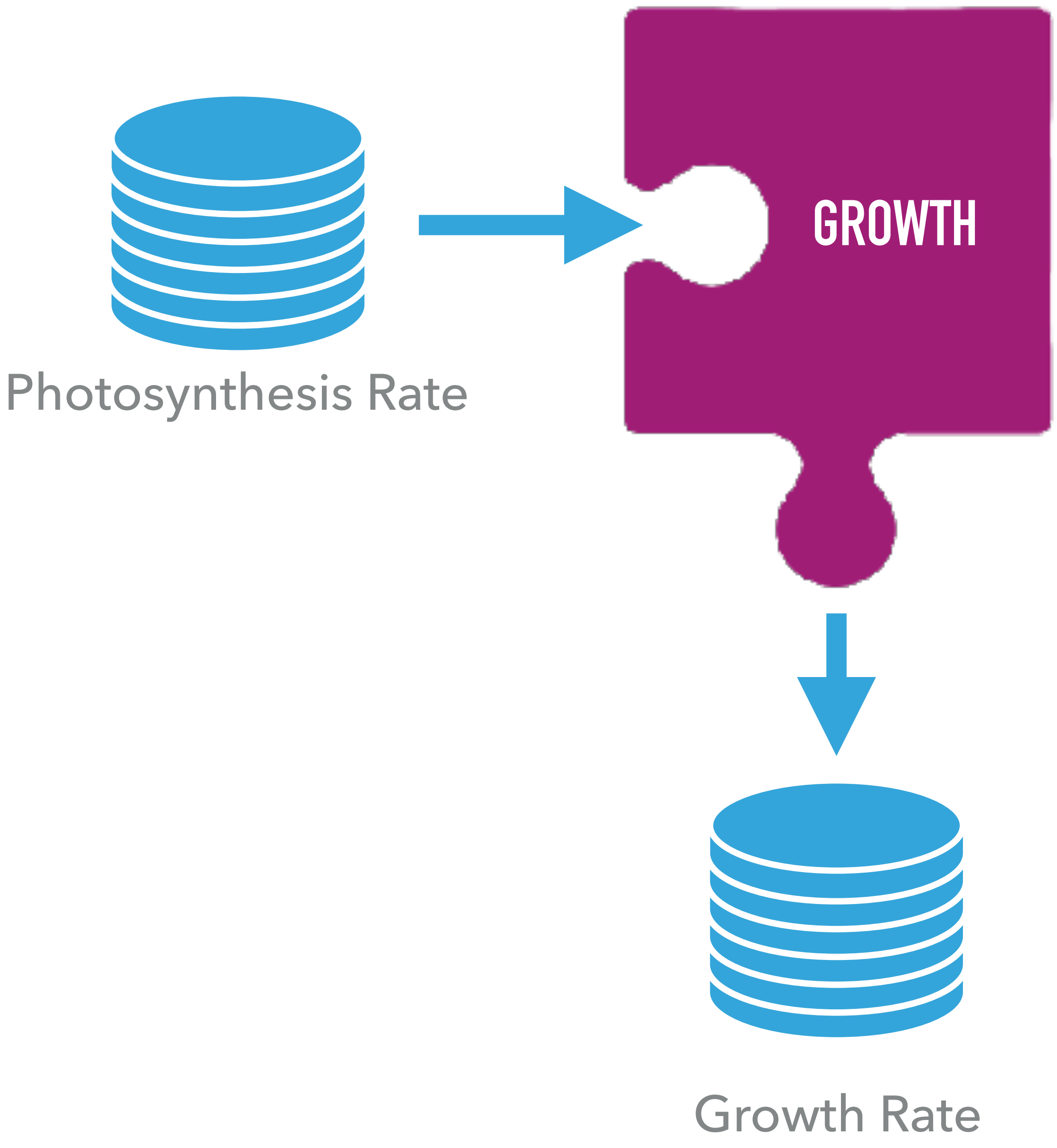
```
connections:
  - input: ./Input/temperature.txt
    output: temperature
    filetype: table
  - input: ./Input/co2.txt
    output: co2
    filetype: table
  - input: ./Input/light_intensity.txt
    output: light_intensity
    filetype: table
  - input: photosynthesis_rate
    output: ./Output/photosynthesis_rate.txt
    filetype: table
    field_names: photosynthesis_rate
```

RUNNING THE PHOTOSYNTHESIS MODEL

```
$ cisrun photosynthesis.yml photosynthesis_to_file.yml
```



GROWTH MODEL

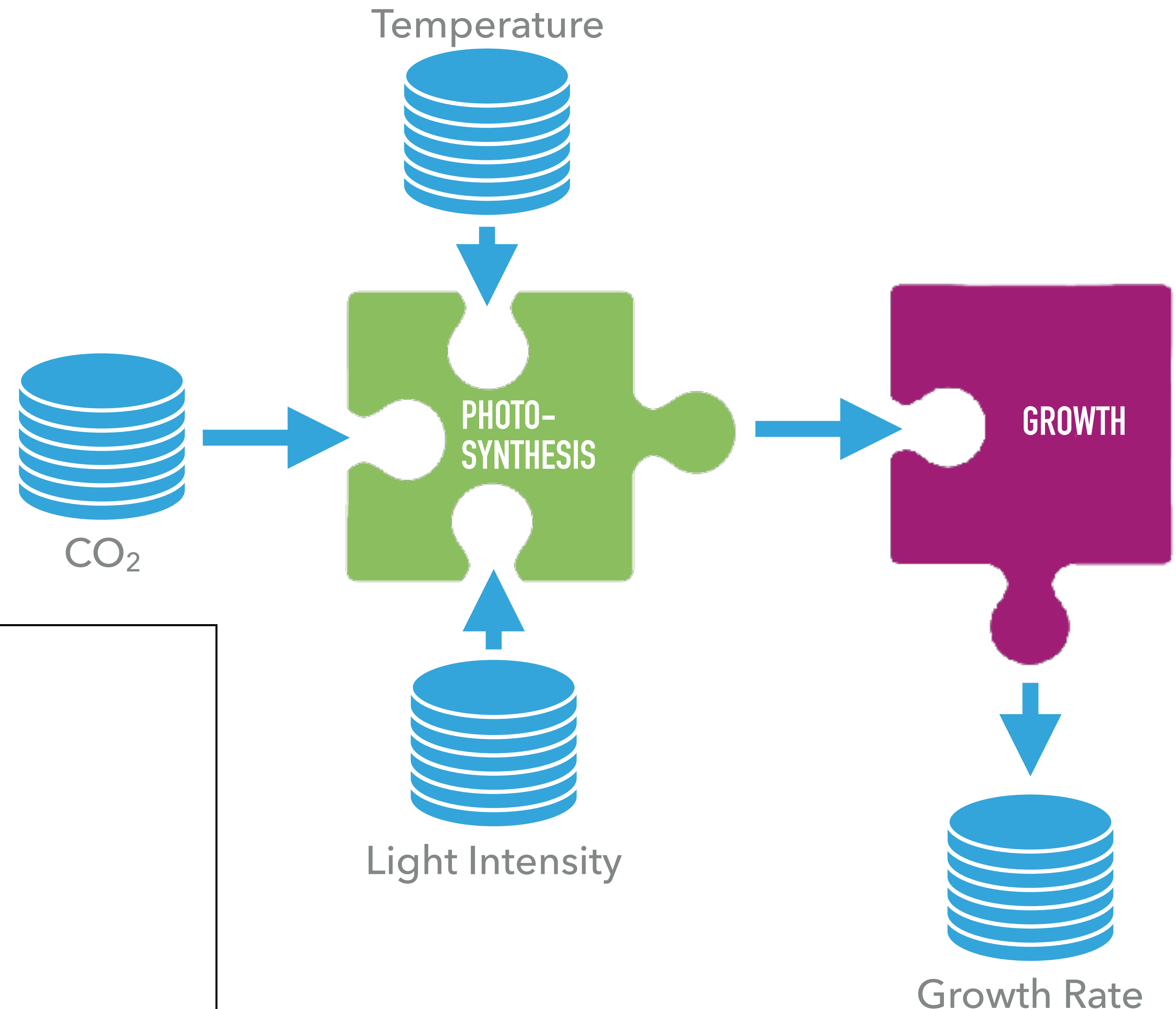


growth.yml

```
1. | model:
2. |   name: GrowthModel
3. |   language: matlab
4. |   args: ./growth_transformed.m
5. |   inputs:
6. |     - growth_photo_rate
7. |   outputs:
8. |     - growth_rate
```

WORKED EXAMPLE: CONNECTION TO ANOTHER MODEL

PHOTOSYNTHESIS + GROWTH MODEL



photosynthesis_to_growth.yml

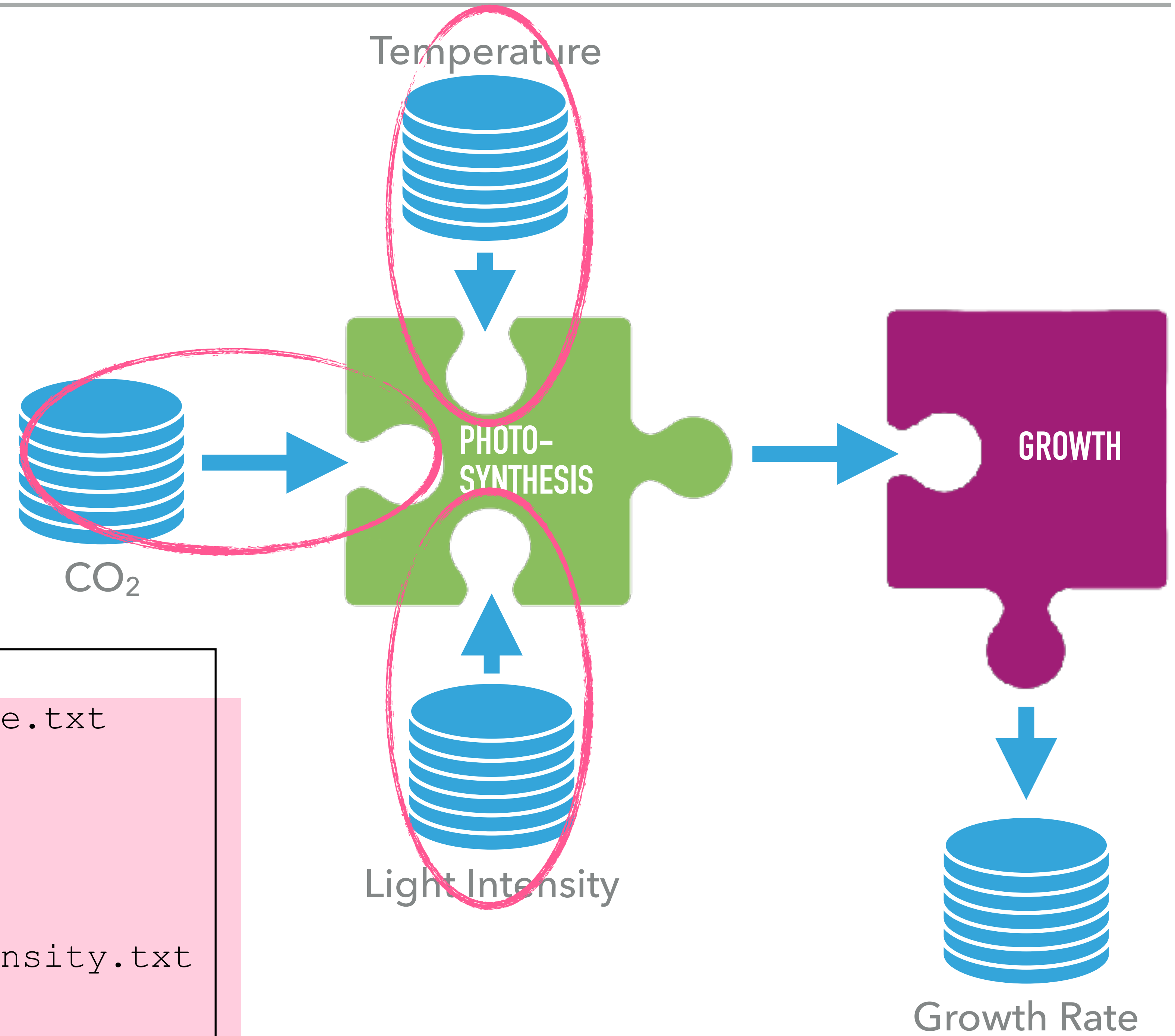
```
connections:
```

PHOTOSYNTHESIS + GROWTH MODEL

Same input files as isolated photosynthesis model.

photosynthesis_to_growth.yml

```yaml
connections:
  - input: ./Input/temperature.txt
    output: temperature
    filetype: table
  - input: ./Input/co2.txt
    output: co2
    filetype: table
  - input: ./Input/light_intensity.txt
    output: light_intensity
    filetype: table
```

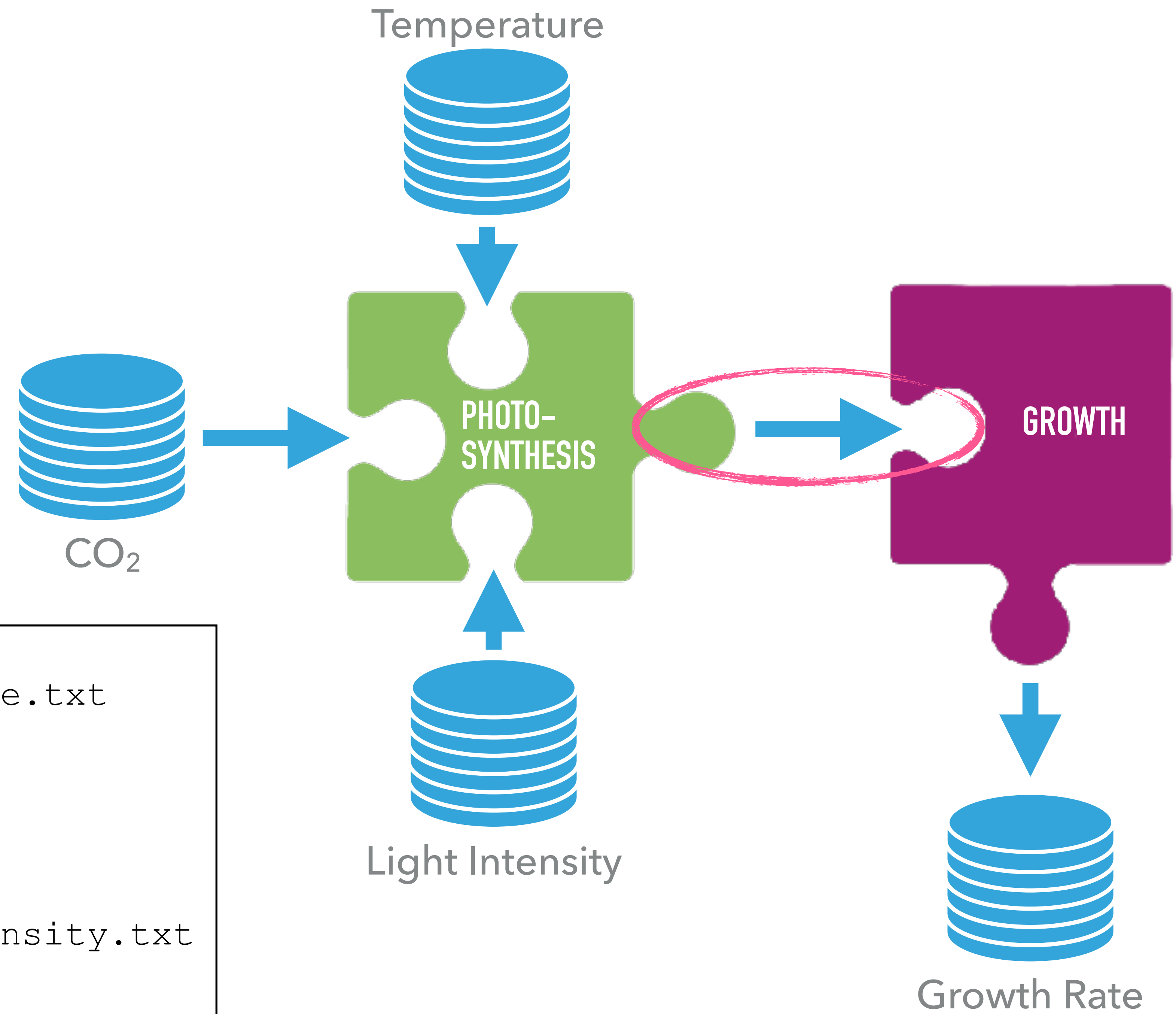


PHOTOSYNTHESIS + GROWTH MODEL

Same input files as isolated photosynthesis model.

photosynthesis_to_growth.yml

```
connections:
  - input: ./Input/temperature.txt
    output: temperature
    filetype: table
  - input: ./Input/co2.txt
    output: co2
    filetype: table
  - input: ./Input/light_intensity.txt
    output: light_intensity
    filetype: table
  - input: photosynthesis_rate
    output: growth_photo_rate
```

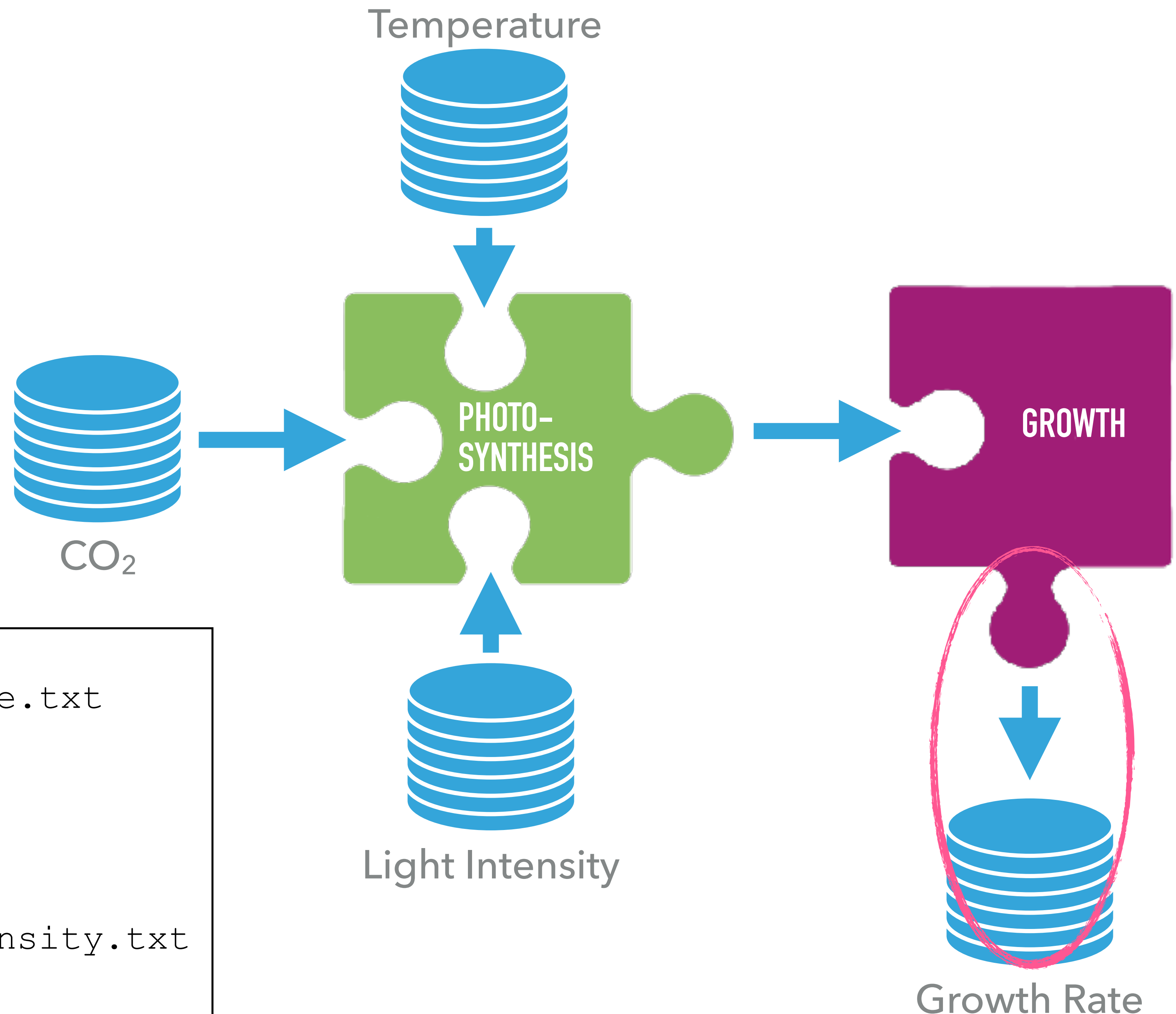


PHOTOSYNTHESIS + GROWTH MODEL

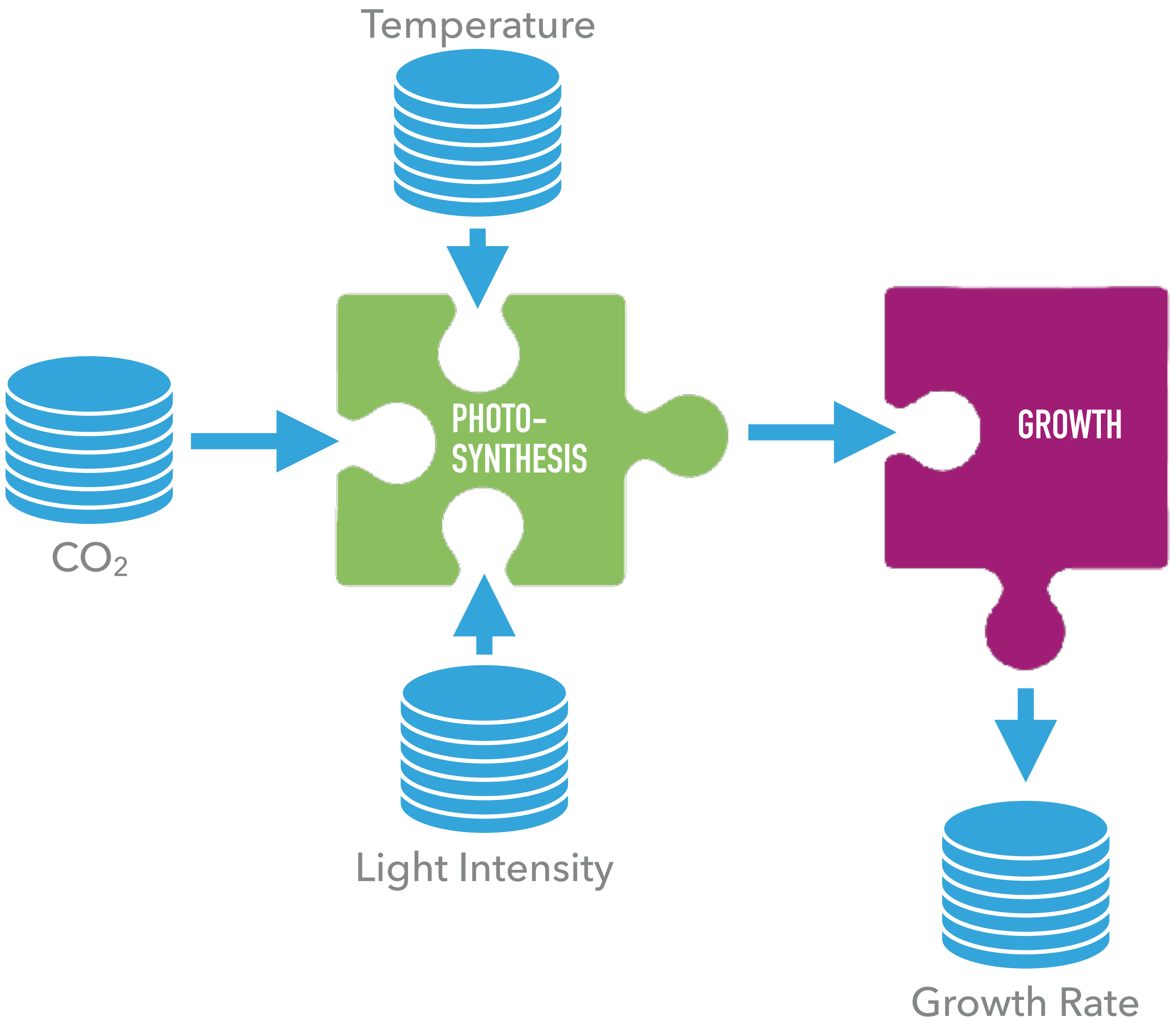
Same input files as isolated photosynthesis model.

photosynthesis_to_growth.yml

```
connections:
  - input: ./Input/temperature.txt
    output: temperature
    filetype: table
  - input: ./Input/co2.txt
    output: co2
    filetype: table
  - input: ./Input/light_intensity.txt
    output: light_intensity
    filetype: table
  - input: photosynthesis_rate
    output: growth_photo_rate
  - input: growth_rate
    output: ./Output/growth_rate.txt
    filetype: table
    fields: growth_rate
```



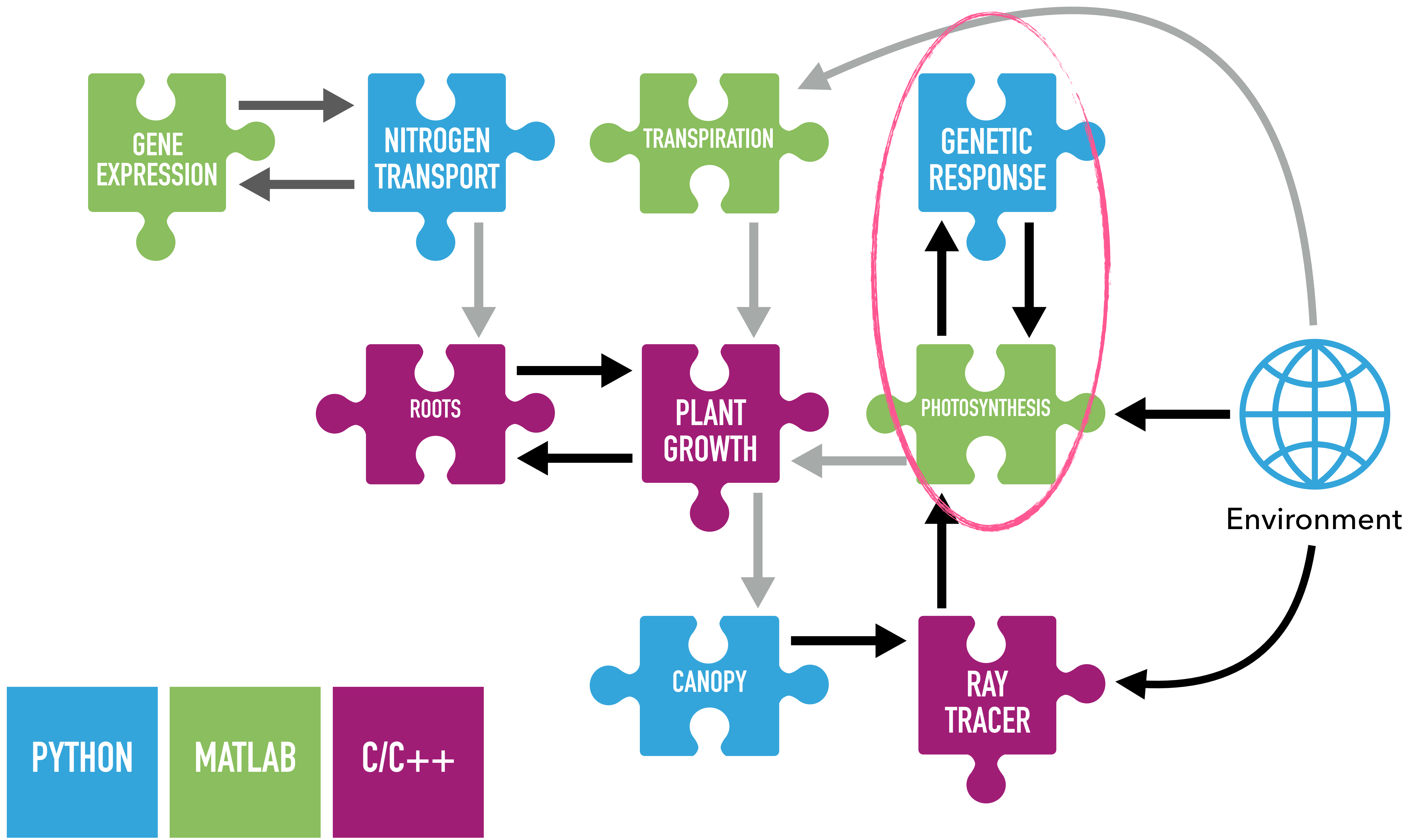
RUNNING THE PHOTOSYNTHESIS + GROWTH MODEL




```
$ cisrun photosynthesis.yml growth.yml
photosynthesis_to_growth.yml
```

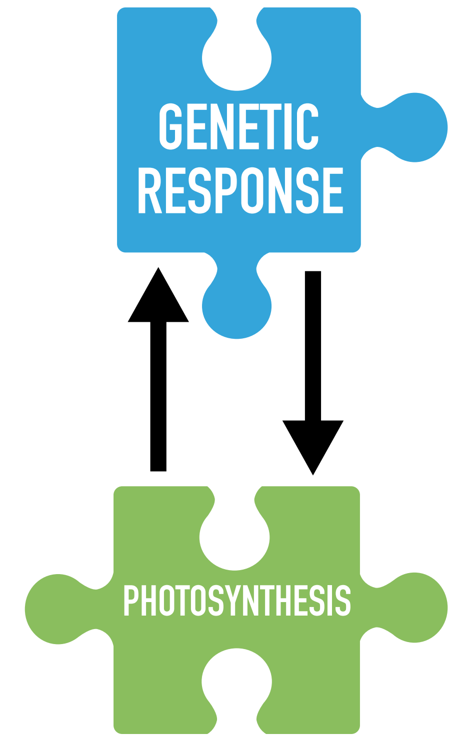
REAL-WORLD EXAMPLE

PLANT MODELS CONNECTED VIA CIS_INTERFACE

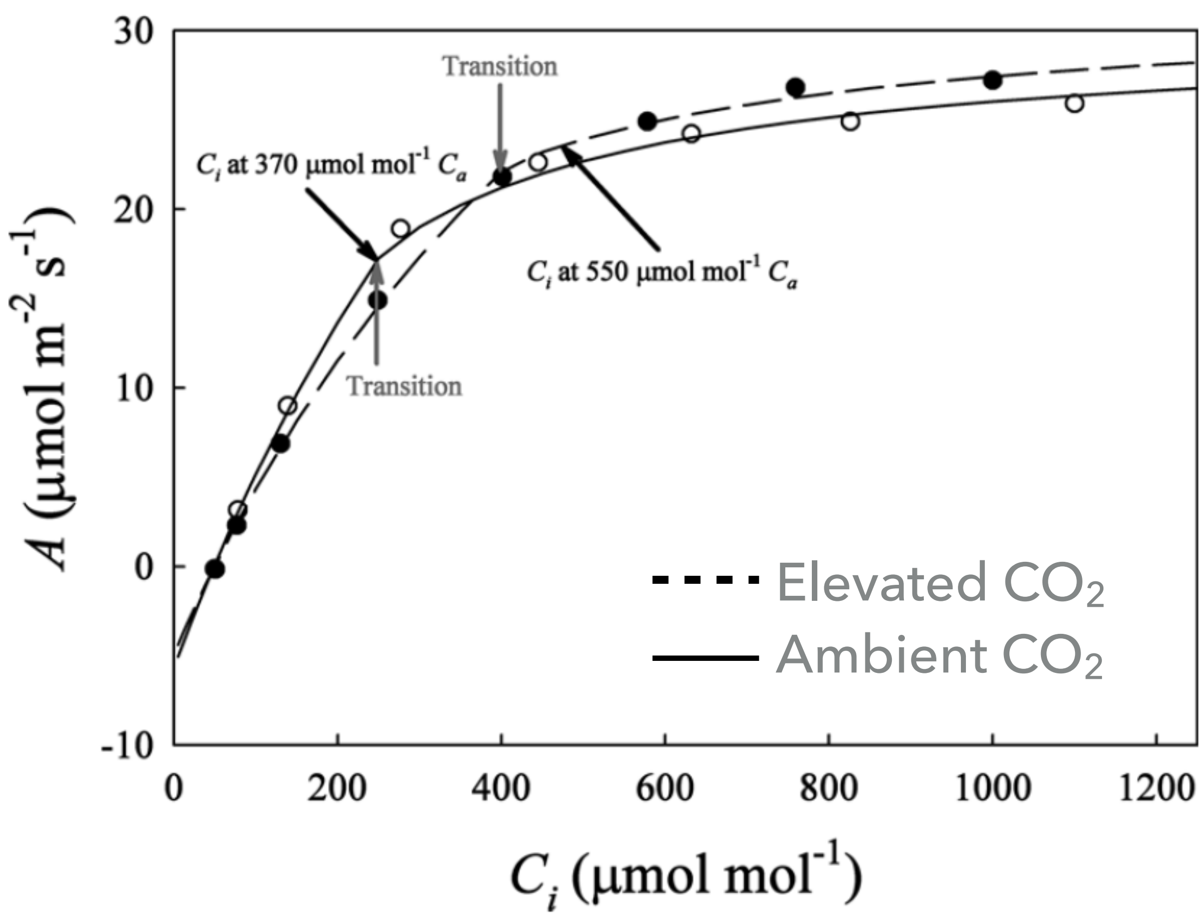


INTEGRATED MODEL REPLICATES CO_2 ACCLIMATION

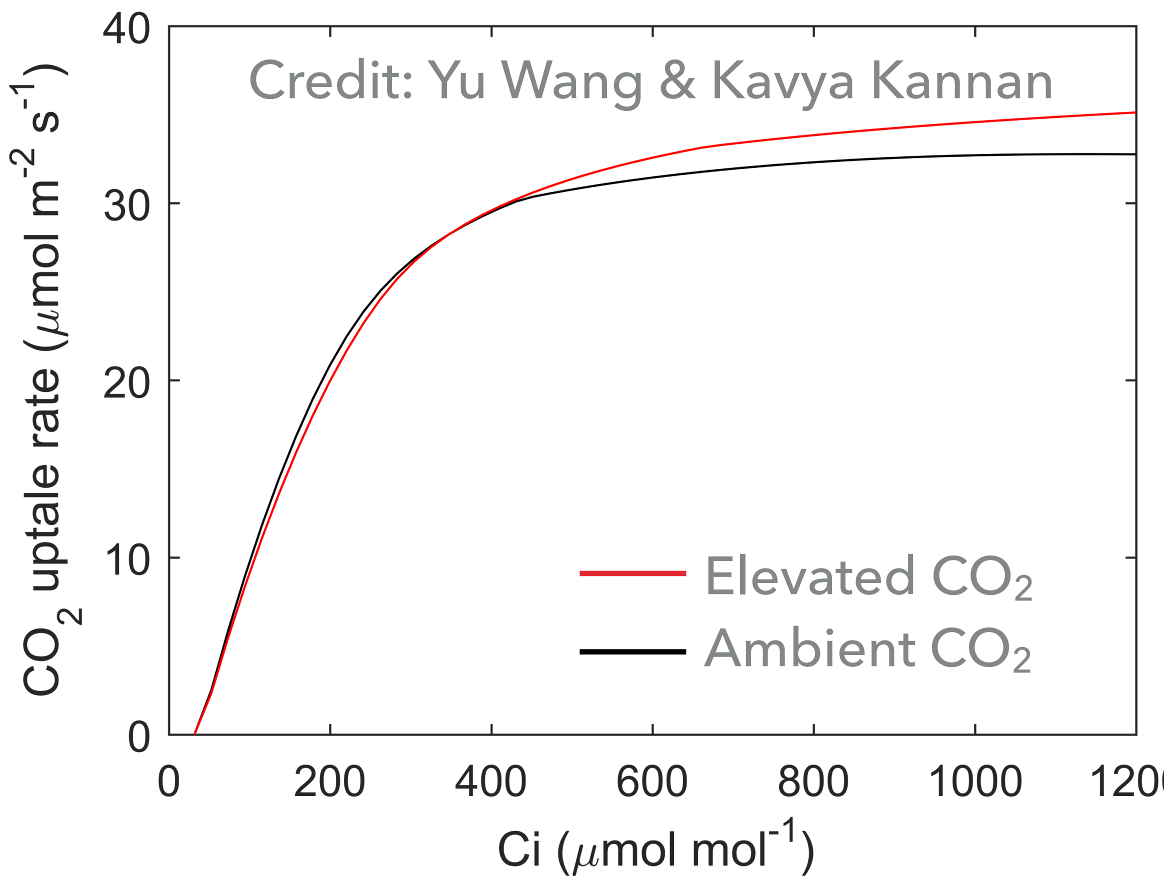
- CO_2 Acclimation: Elevated CO_2 causes decreased photosynthetic efficiency in experiments
- Photosynthesis model alone can reproduce this phenomenon, but does not provide insight into why



Observation



Integrated Models



TECHNICAL DETAILS

CAPABILITIES & IMPLEMENTATION

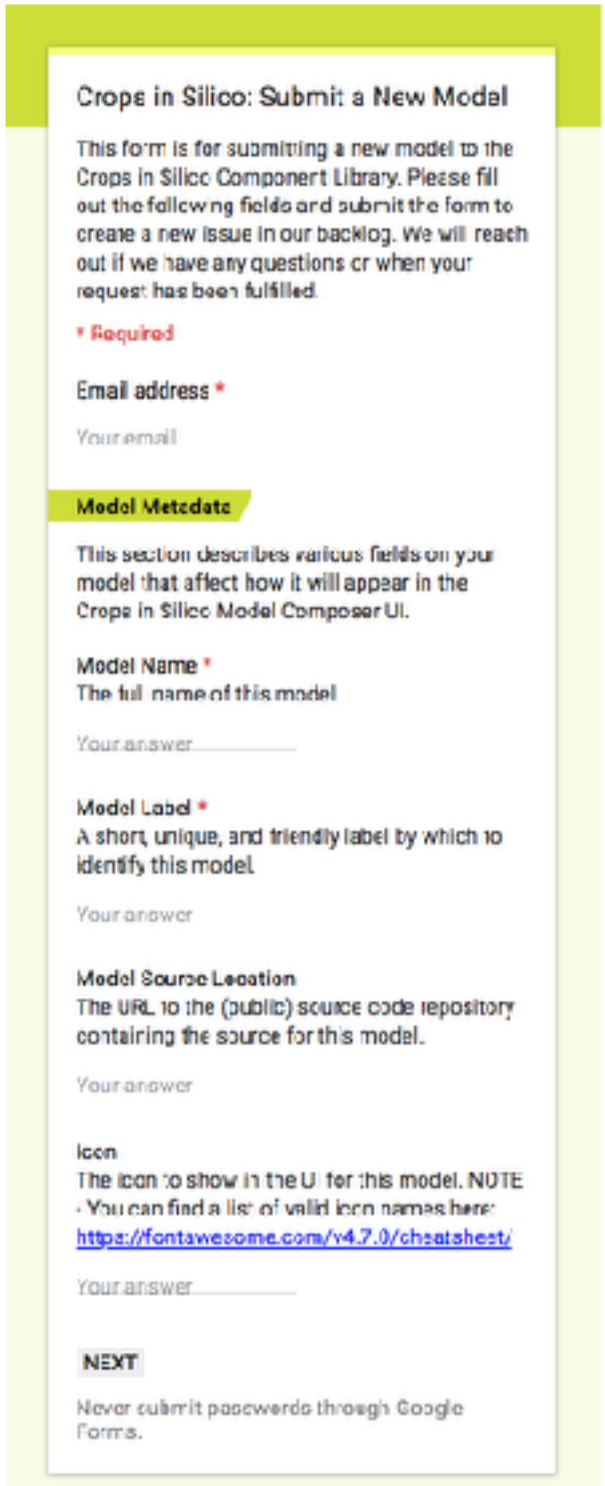
- Public (pip installable) Python package "cis_interface"
- Connect models written in Python, Matlab, C, & C++
- Read/write data from/to formats including CSV, tab-delimited tables, Python pickles, Obj, Ply, .mat
- Communication via Sys V IPC queues, ZeroMQ, or RabbitMQ
- Units via pint, moving to unyt (Nathan Goldbaum)
- Python 2.7, 3.4, 3.5, & 3.6
- Linux, Mac OSX, & Windows

PACKAGE HEALTH

- 100% test coverage
- Continuous integration on Linux, Mac OSX & Windows via TravisCI & Appveyor
- Documentation with automated inclusion of docstrings for API calls in all supported languages via Sphyinx, Doxygen & Breathe
- Tested examples of use cases in all supported languages

CURRENT/FUTURE DEVELOPMENT

MODEL REPOSITORY WITH SUBMISSION FORM


Crops in Silico: Submit a New Model

This form is for submitting a new model to the Crops in Silico Component Library. Please fill out the following fields and submit the form to create a new issue in our backlog. We will reach out if we have any questions or when your request has been fulfilled.

* Required

Email address *

Your email

Model Metadata

This section describes various fields on your model that affect how it will appear in the Crops in Silico Model Composer UI.

Model Name *

The full name of this model

Your answer

Model Label *

A short, unique, and friendly label by which to identify this model

Your answer

Model Source Location

The URL to the (public) source code repository containing the source for this model.

Your answer

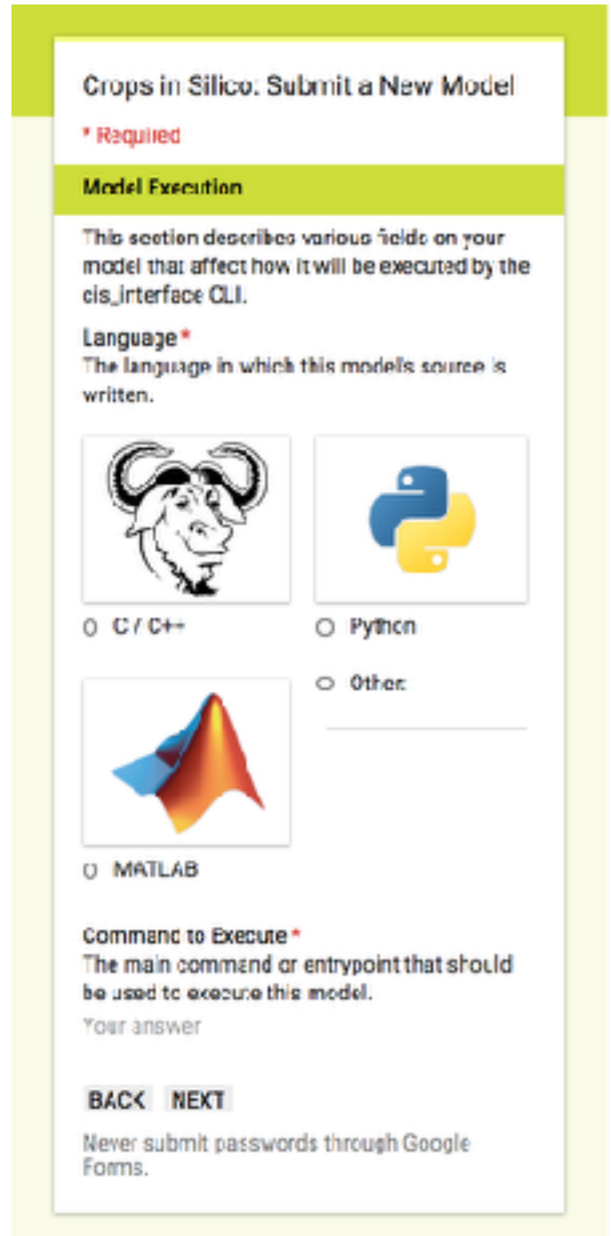
Icon

The icon to show in the UI for this model. NOTE - You can find a list of valid icon names here: https://fontawesome.com/v4.7.0/cheatsheet/

Your answer

NEXT

Never submit passwords through Google Forms.


Crops in Silico: Submit a New Model

* Required

Model Execution

This section describes various fields on your model that affect how it will be executed by the cis_interface CLI.

Language *

The language in which this models source is written.

C / C++

Python

MATLAB

Other:

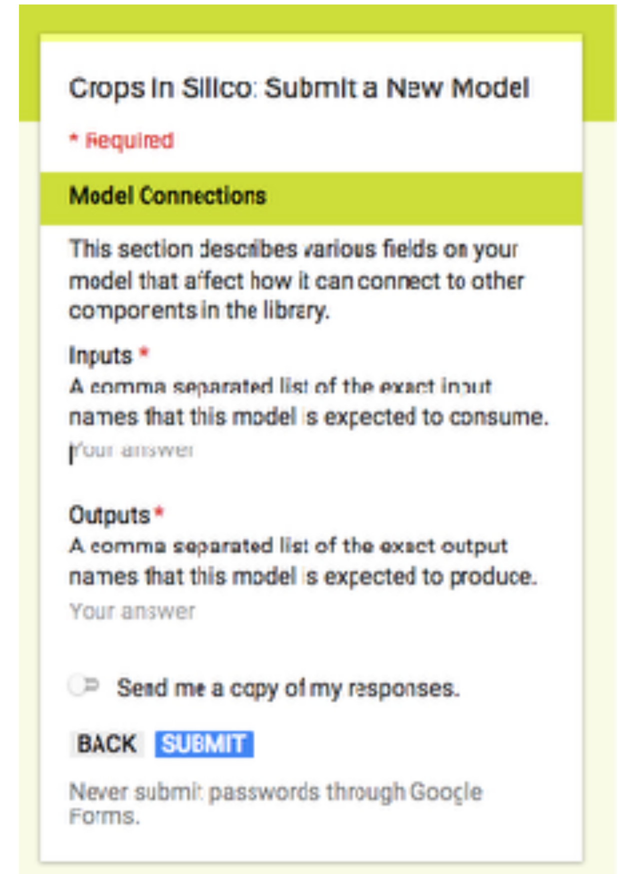
Command to Execute *

The main command or entrypoint that should be used to execute this model.

Your answer

BACK NEXT

Never submit passwords through Google Forms.


Crops in Silico: Submit a New Model

* Required

Model Connections

This section describes various fields on your model that affect how it can connect to other components in the library.

Inputs *

A comma separated list of the exact input names that this model is expected to consume.

Your answer

Outputs *

A comma separated list of the exact output names that this model is expected to produce.

Your answer

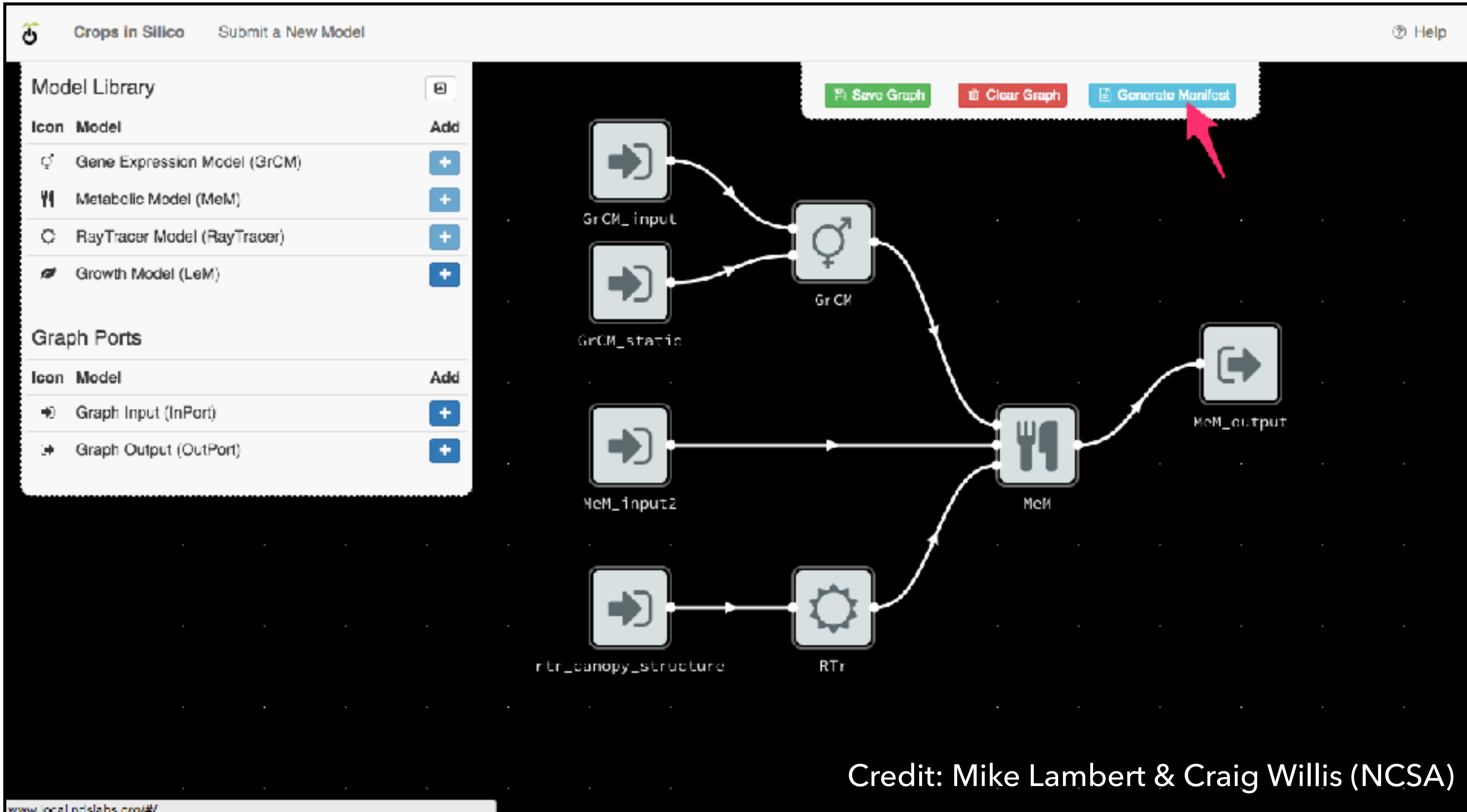
Send me a copy of my responses.

BACK SUBMIT

Never submit passwords through Google Forms.

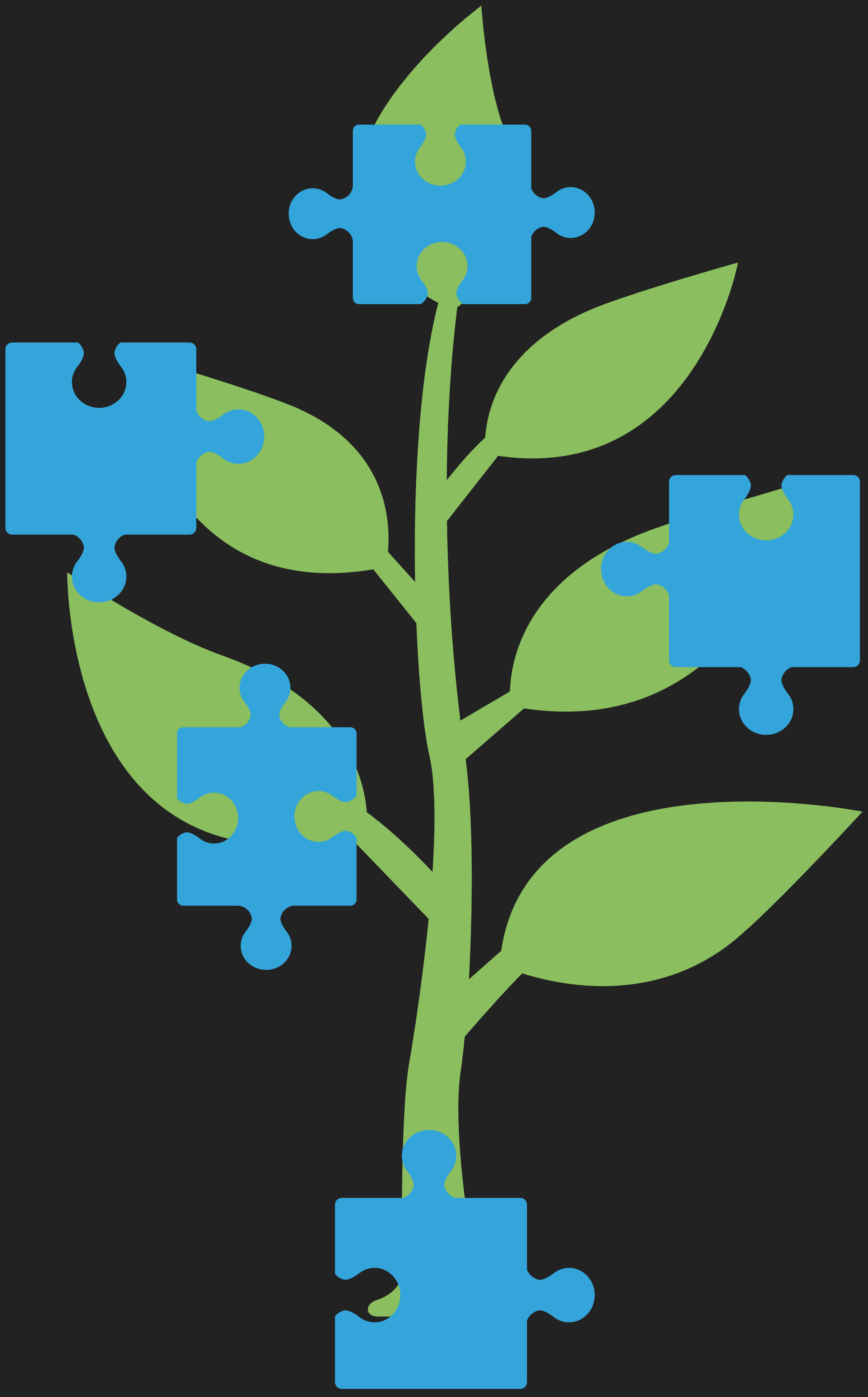
Credit: Mike Lambert & Craig Willis (NCSA)

USERS CREATE NETWORKS VISUALLY, GET YAML BACK



MORE MODELS, MORE SCIENCE

- Connect models written in R, Fortran & Java
- Connect Matlab models using Octave to eliminate need for a Matlab license
- Domain specific data formats (e.g. genetic regulatory networks)
- Tools for running on distributed compute resources (HPC clusters & cloud compute)
- Validation/suggestion of connections via units
- Automated aggregation & transformation of data
- Control flow for models (loops & conditionals)



CIS_INTERFACE

BUILT FOR PLANTS, BUT…

NOT SPECIFIC TO PLANTS

DO YOU HAVE A MODEL?

Github: https://github.com/cropsinsilico/cis_interface

Docs: https://cropsinsilico.github.io/cis_interface/

Project Website: http://cropsinsilico.org/