Introduction to pyCOT

Tomás Veloz

1 Introduction

PyCOT is a library to apply chemical organization theory for modeling and analysis purposes. First algorithms aimed to compute organizations and self-maintaining sets were developed by the group of Peter Dittrich between 2007 and 2010. The idea of developing a library was conceived by Tomas Veloz, Pedro Maldonado and Alejandro Bassi in Santiago, Chile, 2017. While Bassi and Maldonado explored how to perform calculations in efficient manners in R and C respectively, the first development in python was pyRN between 2020 and 2022 as part of the Templeton grant "Origins of Goal Directedness", and is available in github[1]. That preliminary version contains some of the important elements of pyCOT, but the complex handling of data structures implied lack in versatility, so Tomas Veloz rebuilt a version of the library using a more efficient class structure, introducing the notion of subRN and supRN that can inherit maximal information, and adding several functions that permit COT analysis, in an effort to integrate and complete the scattered previous developments.

2 Understanding the pyCOT object

A general COT object of analysis is a part of a reaction network $(\mathcal{M},\mathcal{R})$ where $(\mathcal{M},\mathcal{R})$ is a set of species and $(\mathcal{M},\mathcal{R})$ is a set of reactions. Note that every pair $(X\subset\mathcal{M},R\subset\mathcal{R})$ are in principle a possible choices for a part of the reaction network. This implies that some routines might require exploration of exponential combinations. For this reason, efficiency is important.

Previous work reveals we encounter three levels of analysis: Relational, Stoichiometric and Dynamical. Interestingly, these require different (and increasingly complex) mathematical structures to define properties.

[1] https://github.com/pmaldona/pyRN

Relational properties require set-theoretical operations such as union, intersection, containment. Stoichiometric properties require linear algebraic manipulations. Dynamical properties require simulations.

2.1 Identification of species and reactions

For this reason, the basic structure of the pyCOT object identifies species and reactions in two different ways:

- SpBt/RnBt: Bitarray identification
- SpStr/RnStr: List of strings (species/reaction names) identification

By default, users shall work at the 'Str' identification level, which is easier to read, but functions generally operate at the bitarray level in the backend for optimization. te

There are a number of functions to get one representation from another. For example, for a given pyCOT object "MR" and a subset of species $X \subseteq \mathcal{M}$ represented as a list of species x we have that "self.get_bt_from_species(x)" gives the bitarray representation of x. So, if $|\mathcal{M}| = n$ then "self.get_bt_from_species(x)" is a a bitarray of length n, whose $i-th$ coordinate is True (1) if $s_i \in X$ and False (0) else.

2.2 Reactions as pairs of vectors

The representation of reactions requires two lists

- RnVecS: Reaction support
- RnVecP: Reaction product

These vectors are defined as numpy.array objects.

The library is built in a way that relational calculations are made at the bitarray (string) identification and stroichiometric calculations are made using the np.array identification. This helps to optimize the speed at which operations are made.

Multiple functions are again developed to deal with the use of reactions in a simple way. For example, "self.get_supp_bt_from_reaction(reaction_name,t)" gives the bitarray representation of reaction "reaction_name" of the species that are support of the reaction in question. Note that there is a parameter t that indicates a threshold, so that if the support vector (RnVecS) at the coordinate of a

given species is larger than the threshold then the species is considered with True in the bitarray returned. Otherwise it gives a False value.

WRITE LIST OF TRANSFORMATIONS

2.3 Relational Important Functions

2.4 Connectivity Relations

There multiple ways to define connectivity relations. The simplest one, and most widely used in COT algorithms is the topological connectivity, meaning that there is a way to link two species through the reactions. s_i and s_j are **immediately connected** iff and only iff there exists a reaction $r \in \mathcal{R}$ such that $\{s_i, s_j\} \subset (r) \cup \prod(r)$. Two species are**connected** if there is a sequence of immediately connected species starting from one species and ending in the other species.

Another interesting connectivity relation expresses whether connectivity in a directional way. That is s_1 and s_2 are **immediately directionally connected** if and only if there is a reaction $r \in \mathcal{R}$ such that $s_1 \in (r)$ and $s_2 \in \prod(r)$. This relation can also be generalized to sequences of reactions leading to the notion of **directionally connected**.