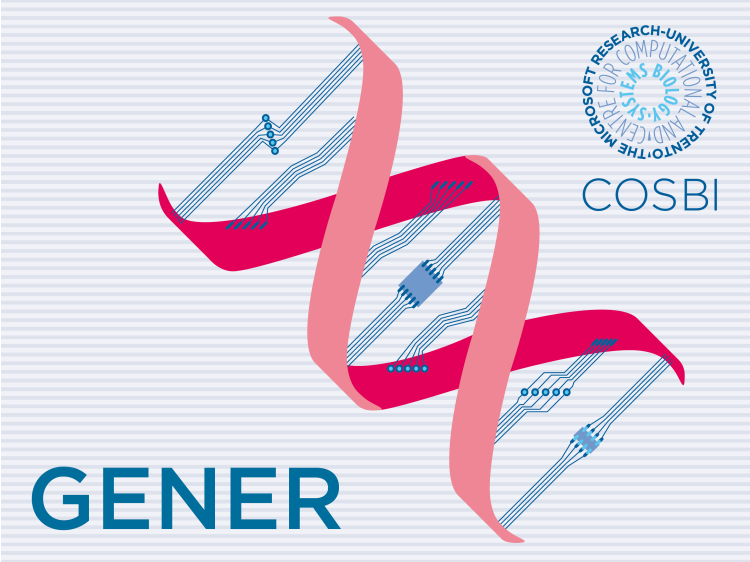

Gener User's Guide

Ozan Kahramanoğulları

The Microsoft Research - University of Trento
Centre for Computational and Systems Biology

gener@cosbi.eu

Version 1.0
Date 08/10/2013

Gener User's Guide

1 Introduction

Gener is a tool for performing computational transitions on two-domain DNA strands based on a strand-displacement algebra [Cardelli, 2013]. This algebra is complete for chemical reactions, thus the DNA-strands implemented in Gener can be used to model chemical reaction networks [Lakin et al., 2013]. Gener allows to perform interactive steps for the application of the reduction rules. Gener also includes a feature for performing exhaustive search on the input algebra to compute the reachable states of DNA strands. The computations can be saved and re-opened. Gener expressons can be exported to Microsoft Research's DSD tool. The reductions computed can be exported to LaTeX. *The tool contains introductory examples in its menu, which should be useful for a quick start.*

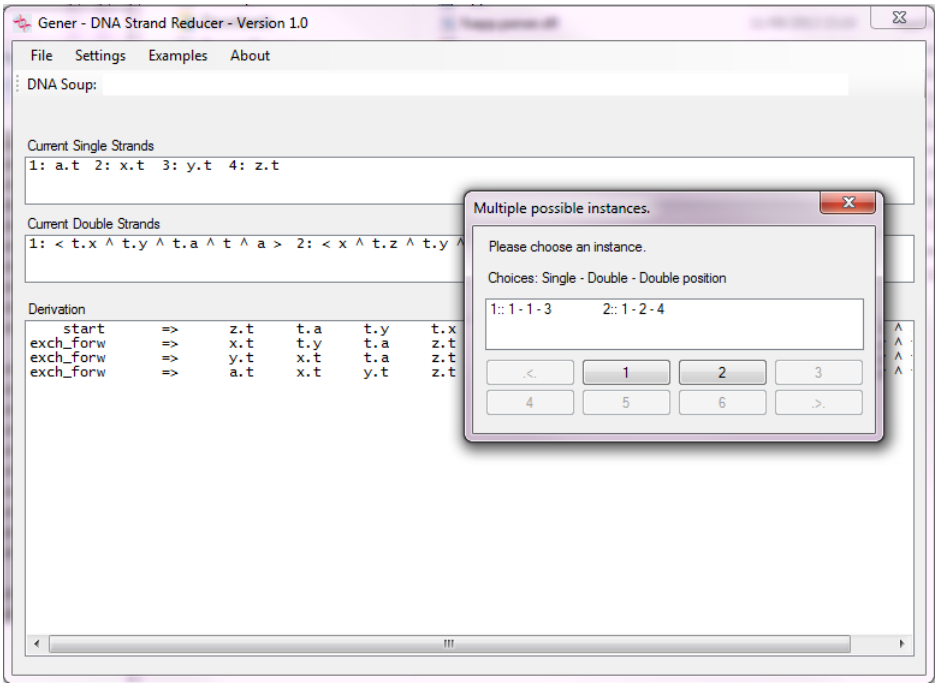

Figure 1: A screenshot of Gener.

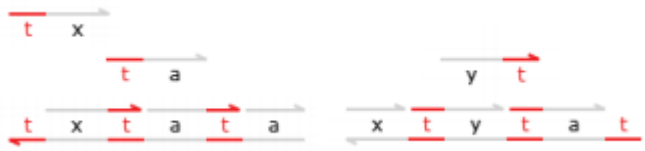

Figure 2: Transducer T_{xy}: initial state and the input `tx`, leading to the reduction $T_{xy} \mid \texttt{tx} \rightarrow \texttt{ty}$.

2 Getting Started

Before using Gener, the user must make sure that Microsoft .NET Framework is installed on the system. This package is available for free from Microsoft web-page.

Gener package is available for download at CoSBi:

`http://www.cosbi.eu/index.php/research/prototypes/gener`

Once the package is downloaded, to run the tool it suffices to unzip the folder at a convenient location and double click the executable.

3 An Example for a Quick Start

We illustrate the tool by a simple example: we type a *DNA soup* into the input field for this. The example below is readily available in the *Examples* menu as *Example 1*. It can thus be loaded by pressing this menu button.

```
t.x t.a  < t ^ x.t ^ a.t ^ a > < x ^ t.y ^ t.a ^ t > y.t
```

This DNA soup consists of three single strands and two double strands, which can be displayed by pressing the *Reduce* button. This DNA soup implements a transducer T_{xy} [Cardelli, 2013], depicted in Figure 2, leading to the reduction `t.x` $\rightarrow$ `t.y` .

When the *Reduce* button is pressed, the rule buttons appear as they become active. This soup can then be reduced by applying the rules below in the following order in order to obtain the output `ty`.

1. `exchange ->`
2. `exchange ->`
3. `exchange <-` (Choice 2)
4. `exchange <-`
5. `right coverage ->`
6. `left coverage ->`

At any point during derivation, the user can press the *backtrack* button to go one step back in the derivation.

The user can save any state of the computation as a latex file, which can be used to print the derivation. The following derivation demonstrates this for the derivation we have just computed.

$\Rightarrow y.t \quad t.a \quad t.x \quad \underline{x^\dagger t.y^\dagger t.a^\dagger t} \quad \underline{t^\dagger x.t^\dagger a.t^\dagger a}$
exch_forw
$\Rightarrow x.t \quad t.a \quad y.t \quad \underline{t.x^\dagger t^\dagger a.t^\dagger a} \quad \underline{x^\dagger t.y^\dagger t.a^\dagger t}$
exch_forw
$\Rightarrow a.t \quad x.t \quad y.t \quad \underline{t.x^\dagger t.a^\dagger t^\dagger a} \quad \underline{x^\dagger t.y^\dagger t.a^\dagger t}$
exch_back
$\Rightarrow t.a \quad x.t \quad y.t \quad \underline{x^\dagger t.y^\dagger t^\dagger a.t} \quad \underline{t.x^\dagger t.a^\dagger t^\dagger a}$
exch_back
$\Rightarrow t.y \quad x.t \quad t.a \quad \underline{x^\dagger t^\dagger y.t^\dagger a.t} \quad \underline{t.x^\dagger t.a^\dagger t^\dagger a}$
right_cov
$\Rightarrow x \quad t.y \quad t.a \quad \underline{x.t^\dagger y.t^\dagger a.t} \quad \underline{t.x^\dagger t.a^\dagger t^\dagger a}$
left_cov
$\Rightarrow a \quad t.y \quad x \quad \underline{t.x^\dagger t.a^\dagger t.a} \quad \underline{x.t^\dagger y.t^\dagger a.t}$

4 The Search Feature

The user can change the settings from 'one-step rule applications' to 'exhaustive search', and this way explore the computation space of the current DNA soup.

A simple search can be initiated switching on the search mode from the settings menu, and then pressing the *Reduce* button. This explores all the possible computations of the current DNA soup. A simple search (without any further options being chosen) displays an enumeration of the available terminal computations of the strand structures by pruning the redundancies in the search space, and prompts a dialog window for the choice of the trace to be displayed. Choosing the all paths option for the search enables the enumeration of the computations with alternative paths, and the variations option includes to the enumeration also those paths with the same terminal as another path, but with an alternative trajectory.

This then displays all the reachable states in a separate window. By choosing the desired state reached in this window, its derivation can be printed on the display.

Along with the search, by choosing the search tree option, the user can display the search tree, and with this, all the intermediate computations are listed in a reserved field. In the search tree graph displayed, different rules are denoted with different colors, which are listed in a legend. By choosing the equal nodes option of the search tree includes in a visualization of the nodes that result in the same DNA strand structures being connected with a dashed gray line in the search tree. The screen shot of an example search tree visualization with Example 1 above is depicted in Figure 3.

5 Gener Rules

Gener implements the following rules as they are given in [Cardelli, 2013].

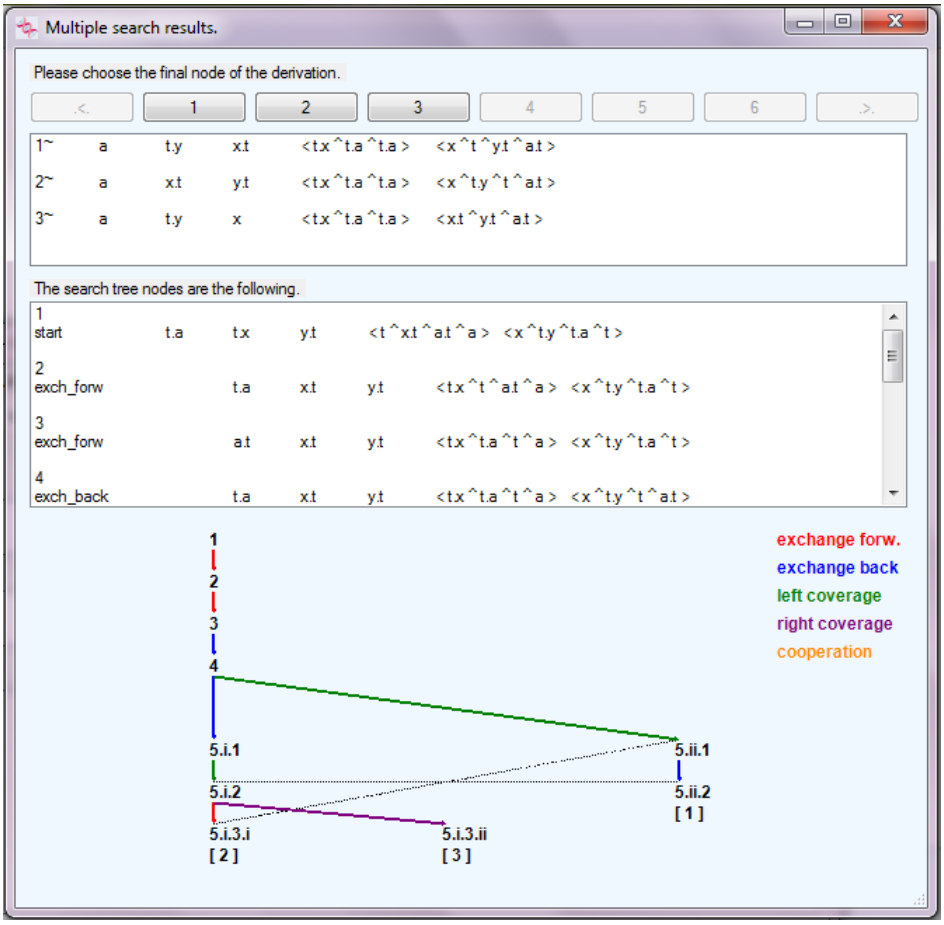

Figure 3: An example screen shot of a search performed on the built in Example 1 of the menu. For the search, 'all paths' option is chosen together with the display options 'search tree' and 'equal nodes'. The gray line between the nodes of the tree denotes the equal nodes.

$\underline{D_1\,{}^\dagger t^\dagger x.t^\dagger D_2} \mid t.x \leftrightarrow \underline{D_1\,{}^\dagger t.x^\dagger t^\dagger D_2} \mid x.t$	Exchange
$\underline{D_1\,{}^\dagger t^\dagger x^\dagger D_2} \mid t.x \to \underline{D_1\,{}^\dagger t.x^\dagger D_2}$	Left Coverage
$\underline{D_1\,{}^\dagger x^\dagger t^\dagger D_2} \mid x.t \to \underline{D_1\,{}^\dagger x.t^\dagger D_2}$	Right Coverage
$\underline{D_1\,{}^\dagger t^\dagger x.y^\dagger t^\dagger D_2} \mid t.x \mid y.t \to \underline{D_1\,{}^\dagger t.x^\dagger y.t^\dagger D_2}$	Cooperation
$\underline{D} \to \emptyset$	Waste
$U_1 \to U_2 \quad \Rightarrow \quad U_1 \mid U \to U_2 \mid U$	Dilution
$U_1 \to U_2 \quad \Rightarrow \quad (\nu x)U_1 \to (\nu x)U_2$	Isolation
$U_1 = U_2, U_2 \to U_3, U_3 = U_4 \quad \Rightarrow \quad U_1 \to U_4$	Well-mixing

References

[Cardelli, 2013] Cardelli, L. (2013). Two-domain dna strand displacement. *Mathematical Structures in Computer Science*, 23(2):247–271.

[Lakin et al., 2013] Lakin, M. R., Phillips, A., and Stefanovic, D. (2013). Modular verification of dna strand displacement networks via serializability analysis. In *Proceedings of the International Conference on DNA Computing and Molecular Programming*, volume 8141 of *Lecture Notes in Computer Science*. Springer.