Biology System Description Language (BiSDL): a modeling language for the design of multicellular synthetic biological systems

Leonardo Giannantoni*, Roberta Bardini†, Alessandro Savino‡ and Stefano Di Carlo§
Politecnico di Torino
Control and Computer Engineering Department
Torino,10129, Italy
Email: *leonardo.giannantoni@polito.it, †roberta.bardini@polito.it, ‡alessandro.savino@polito.it, §stefano.dicarlo@polito.it

***Abstract*—Background: The Biology System Description Language (BiSDL) is an accessible, easy-to-use computational language for multicellular synthetic biology. It allows synthetic biologists to represent spatiality and multi-level cellular dynamics inherent to multicellular designs, filling a gap in the state of the art. Developed for designing and simulating spatial, multicellular synthetic biological systems, BiSDL integrates high-level conceptual design with detailed low-level modeling, fostering collaboration in the Design-Build-Test-Learn cycle. BiSDL descriptions directly compile into Nets-Within-Nets (NWNs) models, offering a unique approach to spatial and hierarchical modeling in biological systems.**

Results: BiSDL's effectiveness is showcased through two case studies on complex multicellular systems: a bacterial consortium and a synthetic morphogen system. These studies highlight the BiSDL proficiency in representing spatial interactions and multi-level cellular dynamics. The language facilitates the compilation of conceptual designs into detailed, simulatable models, leveraging the NWNs formalism. This enables intuitive modeling of complex biological systems, making advanced computational tools more accessible to a broader range of researchers.

Conclusions: BiSDL represents a significant step forward in computational languages for synthetic biology, providing a sophisticated yet user-friendly tool for the design and simulation of complex biological systems with an emphasis on spatiality and cellular dynamics. Its introduction has the potential to transform research and development in synthetic biology, allowing for deeper insights and novel applications in understanding and manipulating multicellular systems.

***Index Terms*—systems biology, synthetic biology, computational biology, domain-specific language**

I. Introduction

Computational methods play a crucial role in synthetic biology, providing powerful tools that significantly improve the design, analysis, and construction of synthetic biological systems, with a particular emphasis on multicellular synthetic systems [1], [2]. These systems implement intricate functions by distributing genetic constructs among different cells [3]. This approach exploits intra and intercellular interactions within the cell population, distributing the metabolic burden to amplify system responsiveness. However, the complexity of these synthetic designs leads to intricate interactions with the host organism, thereby diminishing predictability and controllability [4], [5]. In this context, synthetic morphogenesis applications pose unique challenges as they strive to govern cellular self-organization, which heavily relies on spatial relationships and interactions among cells in space [6].

Computational methods have a key role in the analysis [7], [8], modeling [9], [10], design [1] and optimization [11], [12], [13] of complex biological processes. In particular, for the analysis and prediction of multicellular synthetic systems dynamics, computational tools must offer instruments for modeling and simulation, accounting for multiple spatial and temporal scales [14]. Computational modeling languages serve as powerful tools in this domain. They must expressively represent the target systems while integrating knowledge from diverse sources [15], thus enhancing our understanding of the Design–Build–Test–Learn (DBTL) cycle [16]. Furthermore, to facilitate interdisciplinary collaboration, these languages must support collaborative development, reproducibility, and knowledge sharing [17], [18].

In computational biology, Domain-Specific Languages (DSLs) serve specific applications (see section II). For instance, the Systems Biology Markup Language (SBML) specializes in biochemical networks [19], NeuroML focuses on the structure and function of neural systems [20], and the Simulation Experiment Description Markup Language (SED-ML) handles procedures for running computational simulations [21]. While these languages excel within their applications, their limited scope and interoperability [22] hamper integration into the multi-level models needed for multicellular synthetic biology. The Infobiotics Language (IBL) addresses interoperability by consolidating modeling, verification, and compilation into a single file, streamlining *in silico* synthetic biology processes and ensuring compatibility with the Synthetic Biology Open Language (SBOL) and SBML frameworks [22]. However, IBL lacks support for describing multicellular synthetic

designs and expressing spatial aspects crucial for synthetic morphogenesis applications [6].

Models of multicellular, spatial biological systems can utilize low-level modeling formalisms [23], [24], [25] or multi-level hybrid models that combine different formalisms across multiple scales [14]. Unfortunately, these powerful tools are primarily accessible to expert users, limiting their availability to experimental synthetic biologists. This paper introduces the Biology System Description Language (BiSDL), a computational language for spatial, multicellular synthetic designs that can be directly compiled into simulatable, low-level models to explore system behavior. BiSDL aims to balance simplicity and intuitive usage for broad accessibility, while its expressive power enables the description of biological complexity in multicellular synthetic systems. Building on preliminary work [26], BiSDL supports flexible abstraction, allowing non-experts to reuse high-level descriptions and experts to manipulate or create low-level models. Additionally, BiSDL supports modularity and composition, facilitating the creation and usage of libraries for knowledge exchange, integration, and reuse in the multicellular synthetic biology DBTL cycle. In this work, the low-level models generated from BiSDL descriptions are based on the Nets-Within-Nets (NWNs) formalism [27], chosen for its capability in multi-level and spatial modeling of complex biological processes [25], [28]. Nevertheless, the language is general enough to be integrated with other low-level formalisms.

The paper is organized as follows: section II summarizes existing computational languages for synthetic biology, section III details the design, syntax, and semantics of BiSDL and its compilation into low-level models using the NWNs formalism. section IV showcases BiSDL capabilities through two case studies on multicellular synthetic systems. Finally, section V summarizes the contributions, highlights open challenges, and outlines future developments.

II. Related work

The COmputational Modelling in BIology NEtwork (COMBINE) initiative coordinates the development of inter-operable and non-overlapping standard languages covering different aspects of biological systems [29], [30], [31], [32]. COMBINE DSLs provide intermediary layers between the user and low-level modeling formalisms. They rely on `.xml` for model description and compile into Ordinary Differential Expressions (ODEs) models, making this mathematical modeling formalism accessible by non-expert users. Some of the COMBINE DSLs specialize in intracellular pathways, such as BioPAX [33], and processes, such as Systems Biology Graphical Notation (SBGN) [34], SBML [35] and CellML [36]. SED-ML [37], [21] exclusively aims at managing simulations of system behavior. NeuroML [20] tackles different biological aspects simultaneously, including spatiality and support simulation management, yet specializes in only neuronal systems. Among COMBINE standards, SBOL [38], [39] targets *in silico* synthetic genetic designs, yet is limited to genetic circuits alone, and does not cover any other biological aspect.

Besides COMBINE standards, several computational languages for systems and synthetic biology exist [40]. Antimony is a text-based definition language that directly converts to the SBML standard employed in Tellurium, a modeling environment for systems and synthetic biology [41]. The Cell Programming Language (gro) [42] is a language for simulating colony growth and cell-cell communication in synthetic microbial consortia. It handles the spatiality and mobility of bacterial cells, internal genetic regulations, and mutual communications. Eugene [43] specifies synthetic biological parts, devices, and systems, mainly focusing on genetic constructs and their expression. Genetic Engineering of Cells (GEC) [44] centers over logical interactions between proteins and genes. GEC programs can be compiled into sequences of standard biological parts for the laboratory application. Genomic Unified Behavior Specification (GUBS) [45] focuses on the cell's behavior as the central entity with a rule-based, declarative, and discrete modeling formalism. gro and GUBS can model the interaction between cells. gro also supports the representation of the spatial organization in a system. However, it is limited to bacterial cells only. Also, neither language is easily accessible to non-expert users. Eugene and GEC focus on genetic circuits only or simple molecular interaction networks for representing and exchanging reusable genetic designs through functional modules, such as Standard Parts [46]. Even when combined in more complex structures, such modules only partly comprise the complexity and hierarchy of interdependent regulations and the role of spatiality in biological multicellular designs. IBL [22] is a DSL for synthetic biology that manages several computational aspects into a single specification, overcoming interoperability issues and ensuring seamless compatibility with SBOL and SBML frameworks. Yet, it currently does not support multicellular synthetic systems or spatial aspects.

To overcome the limitations of existing solutions in biological scope, expressivity of multicellular and spatial aspects, and accessibility, BiSDL provides high-level descriptions of intra- and inter-cellular mechanisms over spatial grids, and their direct compilation into low-level simulatable models for the exploration of system behavior.

III. Methods

To help synthetic biologists create models of multicellular synthetic systems, BiSDL is designed for users with varying computational skills, making biological knowledge readable and writable. BiSDL stands at an abstraction level parallel to biological concepts used in experimental science, bridging the gap between these concepts and the more intricate low-level models. BiSDL descriptions combine user-friendliness with the capacity to capture system

complexity. Its development is centered on domain-specific terminology and the ability to compile into NWNs simulation models, discussed in Section III-C. The language syntax covers process hierarchies, spatial relations, and cellular interactions at the intercellular and intracellular levels. While the BiSDL supports the description of the system, the BiSDL models can be compiled into complex NWNs models for the simulation and analysis of system behavior.

A. Biological perspectives and levels of abstraction

The BiSDL syntax supports the description of spatial and multi-level biological concepts through multiple domains and abstraction levels, as illustrated in the Y Chart reported in Figure 1.

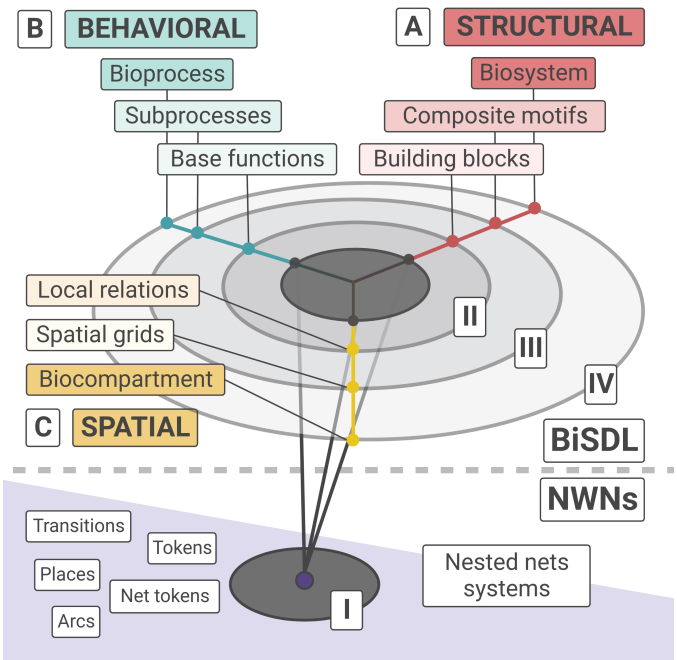


Fig. 1. A scheme of BiSDL domains and abstraction levels inspired to the VHDL Y-Chart [47]. The upper panel (BiSDL) shows the (A) Structural, (B) Behavioral, and (C) Spatial domains and the corresponding high abstraction levels (II, III, and IV) in BiSDL descriptions. The lower panel (NWNs) illustrates the three domains at the low abstraction level relative to the Nets-Within-Nets formalism (I).

Inspired by the Very High-Speed Integrated Circuit Hardware Description Language (VHDL) [47], the BiSDL Y Chart adapts the three description domains defined for VHDL (i.e., Behavioral, Structural, and Physical) to the biological semantics.

The *Structural* domain, illustrated in Figure 1.A, delves into the architecture of biological structures such as transcriptional machinery, protein complexes, or synthetic genetic constructs within a host. On the other hand, the *Behavioral* domain (Figure 1.B) focuses on describing the dynamic functioning, interactions, and transformations of biological elements, encompassing processes like gene transcription, diffusion, and protein degradation. Lastly, the *Spatial* domain (Figure 1.C) outlines the spatial substrate influencing interactions among the elements composing the system (e.g., the spatial organization of a group of cells).

The BiSDL describes each domain at four different abstraction levels, spanning from general biological concepts (high abstraction) to the NWNs modeling formalism elements (low abstraction).

Level IV (Figure 1.IV) describes a *Biosystem* comprising multiple composite motifs within the structural domain. This corresponds, for instance, to a *Bioprocess* made up of multiple bioprocesses in the behavioral domain and a *Biocompartment* defined by multiple spatial grids in the spatial domain. *Level III* (Figure 1.III) elucidates *Composite motifs*, i.e., combinations of building blocks representing complex biological structures in the structural domain. These correspond to *Subprocesses* that emerge from interlaced base functions in the behavioral domain, necessitating *Spatial grids* in the spatial domain to model the underlying spatial relations. *Level II* (Figure 1.II) defines *Building blocks*, capturing fundamental biological concepts in the structural domain. These correspond to *Base functions* in the behavioral domain and simple *Local relations* in the spatial domain. *Level I* (Figure 1.I) describes NWNs formalism elements combined in low-level models of the system.

When describing composite systems (e.g., a biological tissue), the BiSDL covers all domains: structural, spatial, and behavioral (as shown in Figure 1.A-C). These descriptions can include cells, extracellular structures, spatial arrangements, and the processes involved, requiring elements from each of the BiSDL domains. However, simpler descriptions may focus on a single domain, such as the transcription process of a gene concentrating on the behavioral domain. Simple `MODULE` definitions can be combined to form more complex descriptions. The syntax and semantics of these composable BiSDL descriptions are detailed in Additional file 1 - Section 1.

B. Syntax and semantics

All BiSDL descriptions respect the template shown in Algorithm 1 based on a hierarchy of `MODULE`, `SCOPE` and `PROCESS` constructs. They start with naming the `MODULE` (line 1). A `MODULE` encapsulates the complete description of a biological system, encompassing structural, behavioral, and spatial aspects. This includes detailing groups of cells, their spatial arrangement on a two-dimensional grid, intracellular processes, and the spatial diffusion mechanisms facilitating intercellular communications. Modules are self-contained and serve as the fundamental units for reusing and composing existing descriptions.

Each `MODULE` consists of a set of `SCOPE` declarations with defined identifiers and spatial coordinates (lines 3-12 and 13) that describe the relevant biological com-

partments within the modeled system and a set of `DIFFUSION` mechanisms (lines 14-15) that model the diffusion of signals among them. The `SCOPE` declarations may incorporate additional communication methods, such as `PARACRINE_SIGNAL` (line 10) and `JUXTACRINE_SIGNAL` (lines 11-12), describing intercellular communication, either through diffusible signals (paracrine) or direct contact (juxtacrine). The `TIMESCALE` of a module (line 2) sets the base pace of the system dynamics.

Each `SCOPE` contains a set of biological `PROCESS` instantiations with defined identifiers (lines 4-8 and 9). The identifiers comprise *base functions* like transcription, translation, and degradation. The `TIMESCALE` of a process (line 6) is a discrete multiplier of the `MODULE` timescale, determining the relative speed at which the process occurs compared to the base module pace. Different `PROCESSE` instances can connect over the same elements: for example, one process might produce a molecule that regulates a base function in another process. The BiSDL emphasizes ease of description. Each `SCOPE` can reuse a `PROCESS` from another `SCOPE` simply by declaring a `PROCESS` with the same `<process_id>`.

```
1  MODULE <module_id>
2      TIMESCALE <int_value>
3      SCOPE <scope_id> <scope_coord>
4          PROCESS <process_id>
5
6              TIMESCALE <int_value>
7              <basic_function> (<param>, ...)
8              <basic_function> (<param>, ...)
               ⋮
9          PROCESS <process_id>
               ⋮
10         PARACRINE_SIGNAL <signal>, ...
11         JUXTACRINE_SIGNAL <signal> -> <scope_id>
12         JUXTACRINE_SIGNAL ...
13     SCOPE <scope_id> <scope_coord>
           ⋮
14     DIFFUSION <scope_id>, <scope_id>, [<signal>, ...]
15     DIFFUSION <scope_id>, <scope_id>, [<signal>, ...]
       ⋮
```

Algorithm 1: BiSDL general template. Each `MODULE` organizes around a set of `SCOPE` definitions. Each `SCOPE` contains a set of `PROCESS` instances describing the behavior of entities in the `MODULE` and a set of `SIGNAL` declarations describing communication mechanisms among entities. `DIFFUSION` mechanisms support communication among `SCOPE` constructs.

As a simple example, Algorithm 2 provides the BiSDL description of the chemical reaction by which two H_2 molecules react with one O_2 molecule to form two H_2O molecules.

The `MODULE`, whose base `TIMESCALE` is `1`, consists of a biological compartment at coordinates `(0, 0)` within the spatial grid (`SCOPE s`, line 3). The `SCOPE` contains a single `PROCESS`, whose base `TIMESCALE` multiplier is `1`, named `reaction`. Here, the entities describing molecu-

```
1 MODULE example
2     TIMESCALE 1
3     SCOPE s (0, 0)
4         PROCESS reaction
5             TIMESCALE 1
6             CUSTOM_BASE_FUNCTION
                ([2*H2_molecule, O2_molecule],
                [2*H2O_molecule])
```

Algorithm 2: BiSDL description of the chemical reaction by which two H_2 molecules react with one O_2 molecule to form two H_2O molecules.

lar hydrogen (`H2_molecule`) and oxygen (`O2_molecule`) transform into water (`2*H2O_molecule`, line 6). Multipliers for `H2_molecule` and `H2O_molecule` specify the proportion the molecules combine, implying a multiplier with unitary value when not indicated. Additional file 1 - Section 1 describes the syntax and semantics of BiSDL base functions in detail.

C. From BiSDL descriptions to NWNs models

The BiSDL supports the system dynamics analysis by means of simulations. This is obtained by compiling BiSDL descriptions into low-level models based on the NWNs formalism.

1) Compilation of NWNs models: NWNs extend the Petri Nets (PNs) formalism to support hierarchy, encapsulation, and selective communication [27], [24], [9], which makes them suitable to model complex biological processes [25], [28], coherently with the design goals of BiSDL (Section III-A). PNs are bipartite graphs where nodes

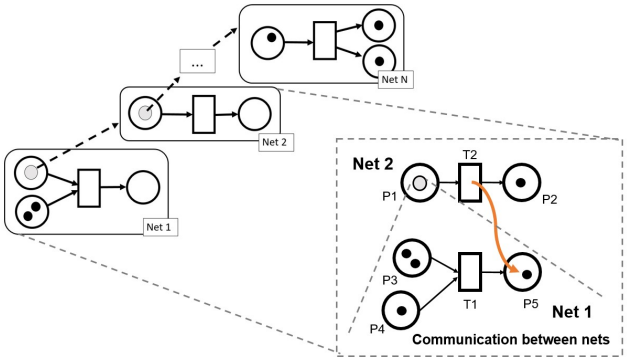


Fig. 2. A representation of the NWNs formalism. Tokens in these PNs can be instances of PNs, thus implementing a hierarchy of encapsulated levels. Channels can interlock two transitions from different nets, allowing the exchange of tokens and information.

can be either *places* or *transitions*. Places represent states the modeled resources can assume. Transitions model the creation, consumption, or transformation of resources in, from, or across places. Tokens model discrete units of resources in different states. Each transition has rules regulating its enabling and activation, depending on the availability of tokens in its input places. Directed arcs link places and transitions to form the desired network architectures. When a transition fires, it consumes the

required tokens from the input places and creates tokens in its output places.

The NWNs formalism is a high-level PNs formalism supporting all features of other high-level PNs: tokens of different types and timed and stochastic time delays associated with transitions. NWNs introduce an additional type of token named *Net Token*. A Net Token is a token that embeds another instance of a PN. With this type of token, NWNs support hierarchical organization, and each layer relies on the same formalism (see Fig. 2). This characteristic introduces the Object-Oriented Programming (OOP) paradigm within the PNs formalism. Therefore, NWNs models express encapsulation and selective communication, allowing the representation of biological compartmentalization and semi-permeability of biological membranes easily. Nets at different levels in the hierarchy evolve independently and optionally communicate through synchronous channels that interlock transitions from different nets, synchronizing their activation upon satisfaction of enabling conditions.

To support NWNs models, BiSDL compilation generates models implemented with `nwn-snakes`, a customized version of the `SNAKES` library [48]. `SNAKES` is an efficient Python library for the design and simulation of PNs [48]. The `nwn-snakes` library presented in this work extends `SNAKES` to handle multi-scale models, ensuring consistency across the hierarchical levels in the model. `nwn-snakes` provides constructs to express the hierarchy of temporal and spatial scales. Every BiSDL `MODULE` in the compiled model is represented by a `Module` class with an individual timescale in, which in turn inherits from the `PetriNet` class implemented in `nwn-snakes`. A prototype BiSDL compiler (`bisdl2snakes.py`) generates Python `Module` classes implementing `nwn-snakes` models from BiSDL descriptions. Detailed instructions on compiler use are available in the BiSDL `GitHub` public repository (see Data Availability384 Statement).

The spatial hierarchy underlying BiSDL descriptions (see Section III-B) is translated into a low-level model based on a system of nested spatial grids represented by PNs. In this model, Places model sub-portions of space allowing the representation of multiple spatial scales. Each Place in a spatial grid can host, as a net token, another spatial grid, ensuring cross-level semantic consistency across different spatial scales. This work provides consistent semantics for two-level hierarchies, which support the intended modeling of multi-cellular systems where both intra- and intercellular mechanisms are described. `nwn-snakes` handles the marking evolution of the two levels synchronously: if marking evolves on one level, the other level mirrors the exact change. Additional file 1 - Section 2 reports the way `nwn-snakes` supports NWNs modeling describing the mapping between BiSDL building blocks and NWNs.

2) Simulation of NWNs models: The exploration of systems dynamics relies on the simulation of `nwn-snakes` models compiled from BiSDL descriptions with the `nwn-petrisim` simulator. The simulator is designed to be simple and easy to use, thus requiring minimal coding as reported in Listing 1.

Listing 1. `nwn-petrisim` instantiation

```
1 test_module = testModule()
2 net_simulator = Simulator(m=test_module,
      output_path=output_path,
      draw_nets=False, mode='exploration')
3 net_simulator.execute(nstep=100)
```

The simulator is instantiated at line 2. The argument `m=test_module` represents the instance of the top-level net to be simulated. The arguments `draw_nets=False` and `mode='exploration'` control the generation of visual output, preventing the creation of images of the net architectures and allowing evolution plots to adapt to the generated output values range.

The simulation, executed in line 3, is discrete, with `nstep=100` determining the number of simulation steps. The simulator analyzes the stochastic evolution of the system. Additionally, it allows simulating the system's response to external stimuli applied to the model as outlined in Listing 2. The simulation comprises a loop running for the specified number of steps (`n_steps`). Within this loop, at every `n` steps, the simulator adjusts the marking of the place that models the stimulus within the network. This adjustment involves adding a random number of black tokens, ranging from 0 to `r`. Subsequently, this modified marking is applied to the simulated model before proceeding to the next simulation step. The values of `n` and `r` control the intensity of the administered stimulus.

Listing 2. `nwn-petrisim` stimuli administration

```
for step in range(n_steps):
    if step % n == 0:
        marking = test_module.get_marking()
        marking['nt']['p'].add(
            [BlackToken()] *
            random.randint(0, r)))
    s.set_initial_marking(marking)
    s.step()
```

In the stochastic simulation supported by `nwn-petrisim`, conflicts among transitions competing for tokens are managed through the randomized ordering of transitions enabling and firing events. All transitions have a user-defined firing probability of `p` set by default to 0.6. Furthermore, for each firing event, the set of tokens consumed by the transition is randomly chosen from those available in the input `place`. These random selections prevent the systematic exclusion of certain transitions from firing.

IV. Results and Discussion

The BiSDL allows synthetic biologists to easily model and design multicellular synthetic systems, simulating their behavior. Synthetic biology aimed first to develop

essential genetic constructs to control specific intracellular processes, then to combine such essential elements into complex circuits within or across cells [49]. Complex circuits enable a broader range of controllable behaviors yet have the drawback of metabolic burden and unknown interactions at the host. To address these constraints, synthetic biology has shifted focus towards designs based on multicellular networks [50], where splitting the overall construct across different cells facilitates integration into host cells and limits their metabolic burden. Construct parts interact *via* intercellular communication, and the desired behavior emerges from the interaction between the different cells. Multicellular synthetic designs must consider complex interactions between the construct and the host cells. Results prove BiSDL capability for (1) accessible model description and (2) exploration of system behavior over two case studies of multicellular synthetic designs: a bacterial consortium (see Section IV-A) and a synthetic morphogen system (see Section IV-B).

A. *Case study 1 - bacterial consortium*

The first case study focuses on implementing gene expression control across various bacterial cells. To achieve this, a synthetic biologist can exploit gene expression regulation across cells operated by the lactose repressor protein (LacI).

Initially, the biologist must identify a reliable source of knowledge regarding synthetic designs that realize the desired behavior. The Registry for Standard Biological Parts holds a collection of predefined genetic constructs with known functionality [51]. These constructs can serve as templates for the DBTL process. By selecting and combining these parts, designing a "bacterial consortium" where the overall genetic device enforces LacI-operated gene expression regulation across cells becomes possible. This consortium comprises two cell types. *Controller cells* establish baseline 3-oxohexanoyl-homoserine lactone (3OC6HSL) production, inhibited by a reference signal (LacI administration) (Figure 3.A). Conversely, *Target cells* initiate Green Fluorescent Protein (GFP) reporter signal production only when receiving the Acyl-homoserine lactones (AHL) molecular signal, which, in this system, is 3OC6HSL (Figure 3.B).

Various Standard Biological Parts contribute to the design. *Part:BBa_C0012* (LacI protein) serves as the reference signal, inhibiting the Lac-repressible promoter. *Part:BBa_I13202* (3OC6HSL Sender Controlled by Lac Repressible Promoter) integrated with S-adenosylmethionine (SAM) and an acylated acyl carrier protein (ACP) substrate to synthesize 3OC6HSL [52] constructs the design in the *Controller cell*. *Part:BBa_E0040* (GFP) together with *Part:BBa_T9001* (Producer Controlled by 3OC6HSL Receiver Device) complete the design with the inducible reporter gene expression in the *Target cell*. The GFP levels serve as the readout signal.

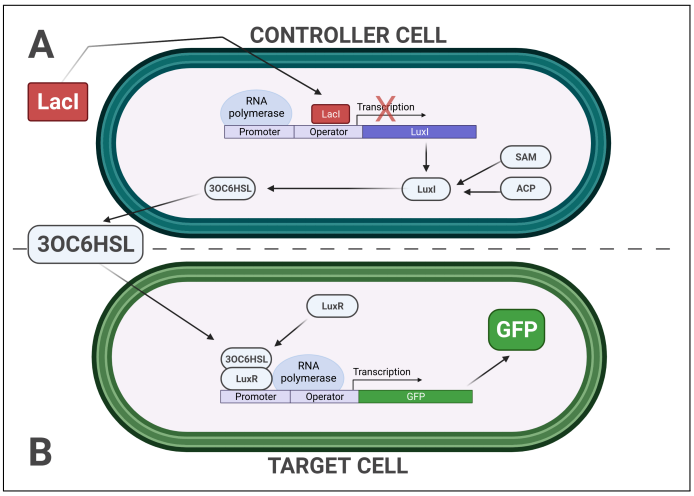


Fig. 3. The multicellular bacterial consortium synthetic design considered for the first case study.

The synthetic biologist who leverages BiSDL to describe the synthetic bacterial consortium should model the synthetic construct split across Controller and Target cells. Moreover, the model must include the biological interactions and mechanisms involved, such as transcriptional processes (gene expression), activation and inhibition of gene expression, protein production and degradation, enzymatic reactions, and inter-cellular signaling. Algorithm 3 provides a BiSDL description of the synthetic bacterial consortium that considers all of these relevant aspects.

The illustrated `bacterialConsortium MODULE` contains two `SCOPE` statements: one for the Controller cell (`Producer`) (Alg. 3, lines 3-20); another one for the Target cell (`Sensor`) (Alg. 3, lines 21-34). Each `SCOPE` definition includes its name and two-dimensional coordinates on the spatial grid underlying the model (Alg. 3, lines 3 and 21). Each `SCOPE` contains a single `PROCESS` representing the fundamental biological functionality of each cell: `AHL_production` for the `producer` and `GFP_production` for the `sensor`. The `TIMESCALE` at the top level (Alg. 3, line 2) indicates the base pace for the `bacterialConsortium MODULE`. On the other hand, the `TIMESCALE` of each `PROCESS` (Alg. 3, lines 5 and 22) indicates the process slowdown factor related to the base pace: `AHL_production` evolves at half the base pace and `GFP_production` evolves at one-third of the base pace. The `bacterialConsortium MODULE` also contains declarations of the `DIFFUSION` processes that set up bidirectional connections between the two `SCOPE` constructs (`producer` and `sensor`) and the diffusion of `AHL_molecule` across them (Alg. 3, lines 33-37).

The BiSDL supports a very compact description of the system, using approximately 25% of the lines of code required by the low-level `nwn-snakes` model: 50 lines of code (see Algorithm 3) versus 203 lines of code in the compiled `nwn-snakes` Python model file (based on the

```
MODULE bacterialConsortium
    TIMESCALE 1
    SCOPE producer (1, 1)
        PROCESS AHL_production
            TIMESCALE 2
            DEGRADATION (LacI_protein)
            DEGRADATION (SAM_molecule)
            TRANSCRIPTION (LuxI_gene, 2*LuxI_mrna,
              INHIBITORS: 5*LacI_protein)
            TRANSLATION (3*LuxI_mrna, 2*LuxI_protein,
              INHIBITORS: 5*LacI_protein)
            DEGRADATION (LuxI_protein)
            TRANSCRIPTION (ACP_gene, ACP_mrna)
            TRANSLATION (ACP_mrna, 4*ACP_protein)
            DEGRADATION (ACP_protein)
            ENZYMATIC_REACTION (3*LuxI_protein,
              [SAM_molecule, ACP_protein], [AHL_molecule])
            ENZYMATIC_REACTION (3*LuxI_protein,
              [SAM_molecule, ACP_protein], [AHL_molecule])
            ENZYMATIC_REACTION (3*LuxI_protein,
              [SAM_molecule, ACP_protein], [AHL_molecule])
            ENZYMATIC_REACTION (3*LuxI_protein,
              [SAM_molecule, ACP_protein], [AHL_molecule])
            ENZYMATIC_REACTION (3*LuxI_protein,
              [SAM_molecule, ACP_protein], [AHL_molecule])
            DEGRADATION (AHL_molecule)
    SCOPE sensor (2, 2)
        PROCESS GFP_production
            TIMESCALE 3
            DEGRADATION (AHL_molecule)
            TRANSCRIPTION (LuxR_gene, LuxR_mrna)
            TRANSLATION (LuxR_mrna, 4*LuxR_protein)
            DEGRADATION (LuxR_protein)
            PROTEIN_COMPLEX_FORMATION
              (3*LuxR_protein, 3*AHL_molecule,
              3*LuxR_AHL_complex)
            PROTEIN_COMPLEX_FORMATION
              (3*LuxR_protein, 3*AHL_molecule,
              3*LuxR_AHL_complex)
            DEGRADATION (LuxR_AHL_complex)
            TRANSCRIPTION (GFP_reporter_gene,
              3*GFP_reporter_mrna, INDUCERS:
              2*LuxR_AHL_complex)
            TRANSLATION (2*GFP_reporter_mrna,
              2*GFP_reporter_protein)
            DEGRADATION (GFP_reporter_protein)
    DIFFUSION producer, sensor, [1*AHL_molecule]
    DIFFUSION producer, sensor, [1*AHL_molecule]
    DIFFUSION producer, sensor, [1*AHL_molecule]
    DIFFUSION producer, sensor, [1*AHL_molecule]
    DIFFUSION producer, sensor, [1*AHL_molecule]
```

Algorithm 3: BiSDL description of the bacterial consortium.

files in the public `GitHub` repository, see Data Availability Statement).

To compile the BiSDL description into a `nwn-snakes` model, the synthetic biologist uses the BiSDL compiler (see Section III-C) to generate a `nwn-snakes` file that contains all the NWNs models required by the BiSDL description. Visualization of the NWNs models relies on the `GraphViz` visualization tool [53], provided by `SNAKES` as a plugin. For this use case, the NWNs description includes a top-level net, where two places contain one net token each, and a bottom level net where these net tokens lie.

Figure 4 visualizes the top-level NWNs model, where the places that contain net tokens correspond to the two BiSDL `SCOPE` statements. The *producer* place holds the *AHL_production* net token (Figure 6), while the *sensor* place holds the *GFP_production* net token (Figure 7). Several transitions connect the two places, allowing the bidirectional diffusion of *AHL_molecule* colored tokens across them and the net tokens they contain during simulation, thanks to the `nwn-snakes` synchronization and communication capabilities (see Section III-C1).

Figures 6 and 7 visualize the *producer* and *sensor* net tokens, respectively. In these PNs, places model genes, transcripts, proteins, and molecules, while transitions model processes involving them, such as transcription, translation, degradation, and enzymatic reactions. Black tokens model discrete quantities of resources in each place and are represented by the `dot` symbol.

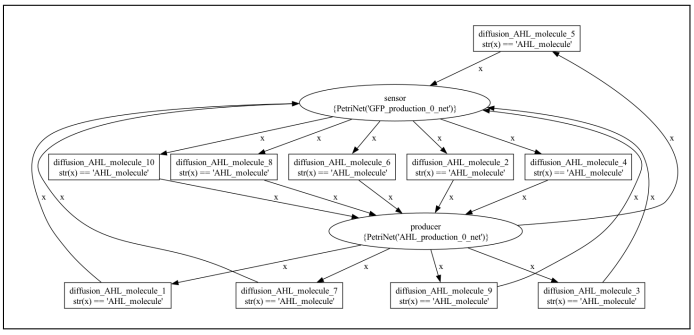


Fig. 4. The top-level bacterial_consortium net architecture.

Simulation of BiSDL-compiled `nwn-snakes` models shows that LacI levels control *GFP_protein* levels, consistently with the expected behavior under the following conditions:

- *noLacI*: the absence of LacI administration;
- *lowLacI*: constant and low LacI administration (`n=3` and `r=3`);
- *highLacI*: constant and high LacI administration (`n=3` and `r=10`);

Values of `n` and `r` determine the intensity of stimulus administration (see Section III-C2).

Figure 5 presents the marking evolution of the `LacI` signal mediators (`Lux1_protein` and `AHL_molecule`) and Target (`GFP_reporter_protein`) in the bacterial consortium after the three consideredLacI administration schemes. *noLacI* (Figure 5.A) does not interfere with `AHL_molecule` levels (Figure 5.D), inducing transcription of high `GFP_reporter_protein` readout signals (Figure 5.G). *lowLacI* (Figure 5.B) hampers `AHL_molecule` levels (Figure 5.E), resulting in the transcription of lower `GFP_reporter_protein` (Figure 5.H). *highLacI* (Figure 5.C) almost shuts down `AHL_molecule` levels (Figure 5.F), suppressing the transcription of high `GFP_reporter_protein` readout signals almost completely (Figure 5.I). The results show consistency with the expected system behavior, that is, an inverse relation between LacI stimulus and readout signal levels.

B. Case study 2 - RGB synthetic morphogen system

The second case study implements a synthetic morphogen system where the spatial interactions and organization of the cells sustain the emergence of a spatial pattern of red, green, and blue (RGB) fluorescent markers. In developmental processes, morphogens transmit positional signals to cells, diffusing from a source to create concentration gradients. Cells interpret these gradients using diverse signaling mechanisms, including paracrine signaling (short-range) and juxtacrine communication (cell-to-cell). Synthetic biology offers the potential

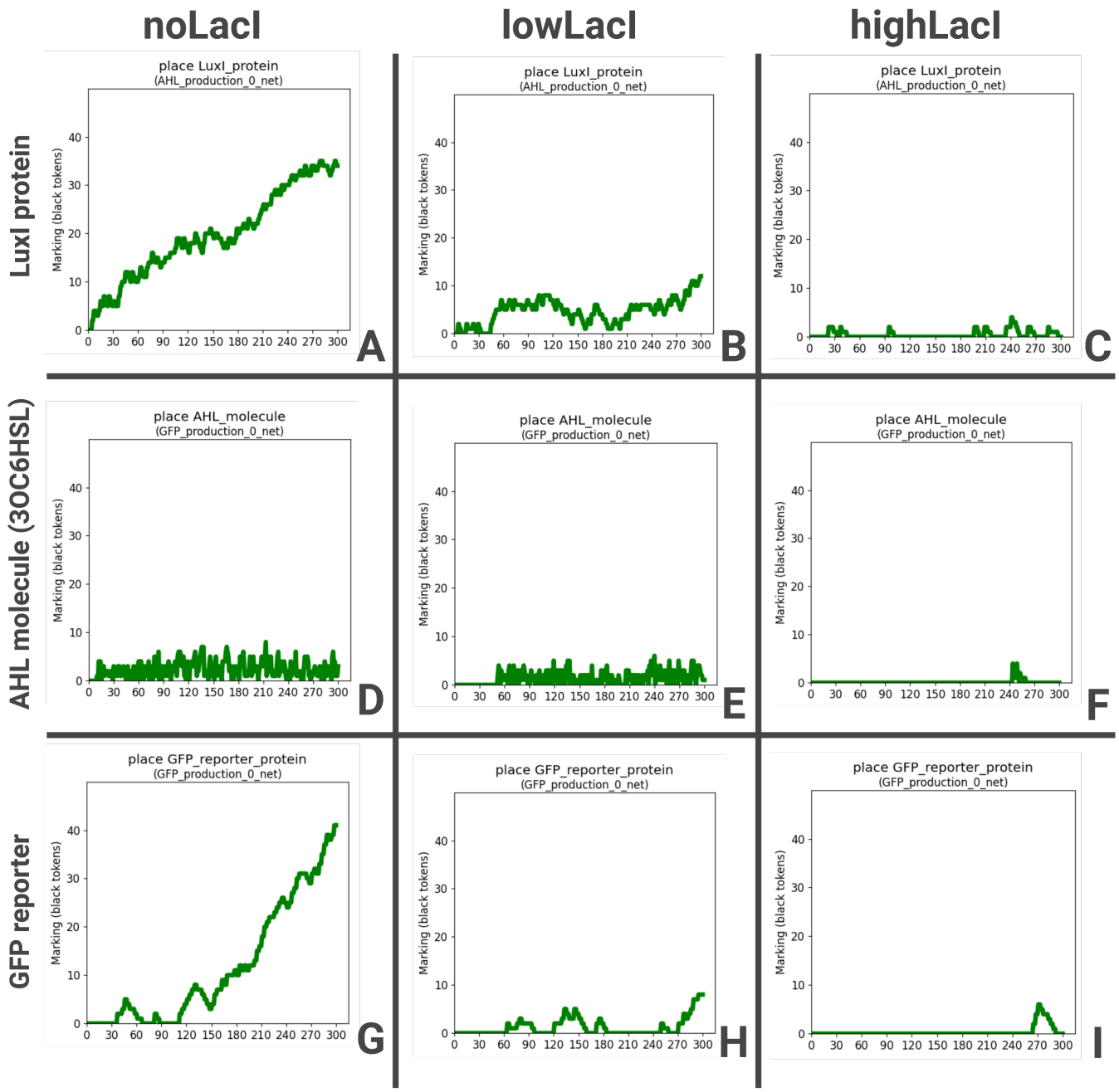


Fig. 5. Marking evolution of the `LacI` signal mediators (`Lux1_protein` and `AHL_molecule`) and Target (`GFP_reporter_protein`) in the bacterial consortium after three LacI administration schemes.

to manipulate these mechanisms, enabling control over spatial arrangement and functional features in synthetic morphogenetic designs. This second case study illustrates how BiSDL descriptions express the essential dynamics underlying a multicellular synthetic design accounting for the role of spatial organization and neighborhood relations among cells.

This time, rather than relying on Standard Parts (refer to Section IV-A), the objective is to replicate designs found in the scientific literature, such as the modular synthetic Notch (synNotch) system outlined in [54], providing a platform for engineering orthogonal juxtacrine signaling, which functions independently of natural cellular communication pathways. It enables specific and controlled cell interactions, featuring an extracellular recognition domain, the Notch core regulatory domain, and an in-

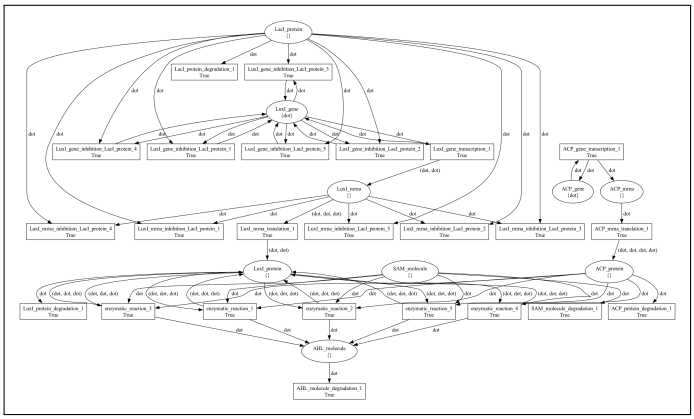


Fig. 6. The producer bacterial_consortium net token architecture.

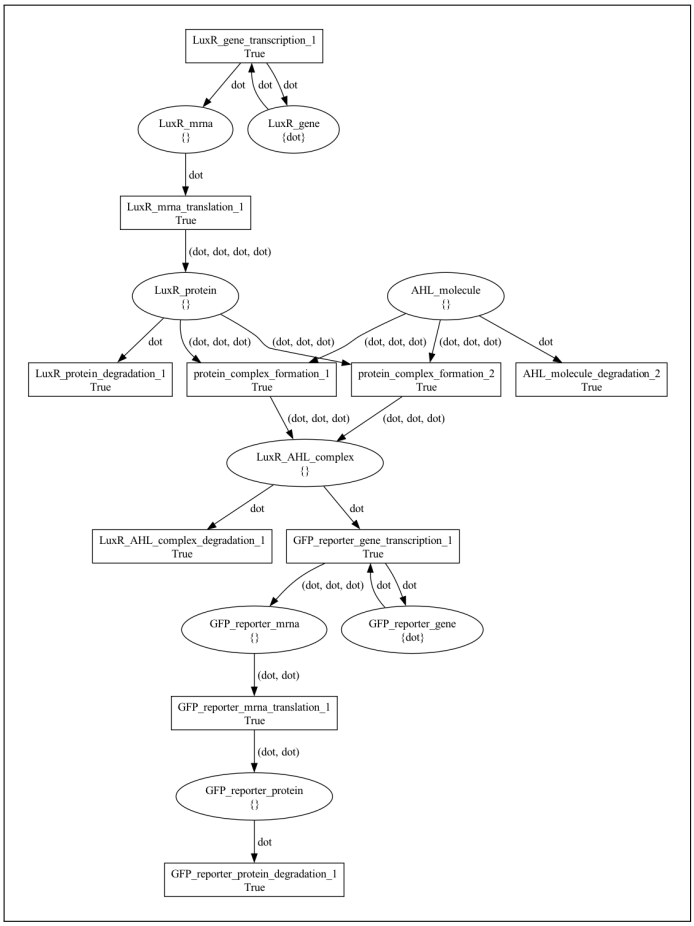


Fig. 7. The sensor bacterial_consortium net token architecture.

tracellular transcriptional domain. With the incorporation of fluorescent markers, the synNotch system proves to be a valuable tool for engineering multicellular synthetic systems.

This second case study comprises a three-layer multicellular circuit in which the Receiver cells (Cells B, Fig. 8.B) inducibly expresses E-cadherin ($Ecad_{hi}$) and a modified form of GFP working as a synNotch ligand on

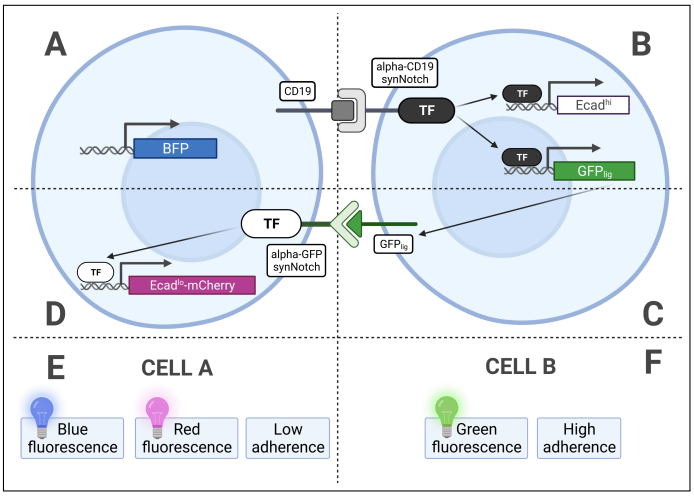


Fig. 8. The multicellular synthetic circuit mediated by cell-cell communication makes the RGB pattern emerge. Adapted from [54].

the cell membrane (GFP_{lig}). Furthermore, GFP_{lig} serves as both a fluorescent reporter and a ligand for a secondary synNotch receptor with the cognate anti-GFP binding domain (Figure 8.C). The Sender cells (Cells A) constitutively express blue fluorescent protein (BFP), CD19 ligand, and the anti-GFP synNotch receptor (Fig. 8.A), which drives expression of a low amount of E-cadherin ($Ecad_{lo}$) fused with a mCherry reporter for visualization (Fig. 8.D).

Cells start as a disorganized aggregate (Fig. 9.A), and CD19 in Cells A activates anti-CD19 synNotch in Cells B, inducing the expression of a high level of E-cadherin ($Ecad_{hi}$) and GFP_{lig} (Fig. 9.B). Cells B thus aggregate and form a compact group in the middle of the aggregate (Fig. 9.C). The GFP_{lig} on Cells B activates the anti-GFP synNotch receptors on Cells A in direct contact with the central group, inducing $Ecad_{lo}$ and the mCherry reporter (Fig. 9.D), and making a spatially organized pattern of cells emerge in a synthetic morphogenetic pattern (Fig. 9.E) with three concentric layers: a green internal core (Cells B expressing $Ecad_{hi}$ and GFP_{lig}) with high cell-cell adhesion, an outer layer of blue cells (Cells A expressing BFP), and a population of red cells in the middle layer (Cells A expressing $Ecad_{lo}$ and mCherry).

Algorithm 4 illustrates the BiSDL description of the RGB synthetic morphogen system, where a central cell induces patterning in its neighbors of the first and second degree. Vertical dots indicate sections describing red and blue cells `SCOPE`s with identical structure as the ones presented but different spatial coordinates. The complete description and the visualization of the compiled `nwn-snakes` PNs models are not included due to their size, but they are available from the BiSDL `GitHub` public repository (see Data Availability statement).

The BiSDL again supports a compact description of the system composed of 165 lines of code, approximately the

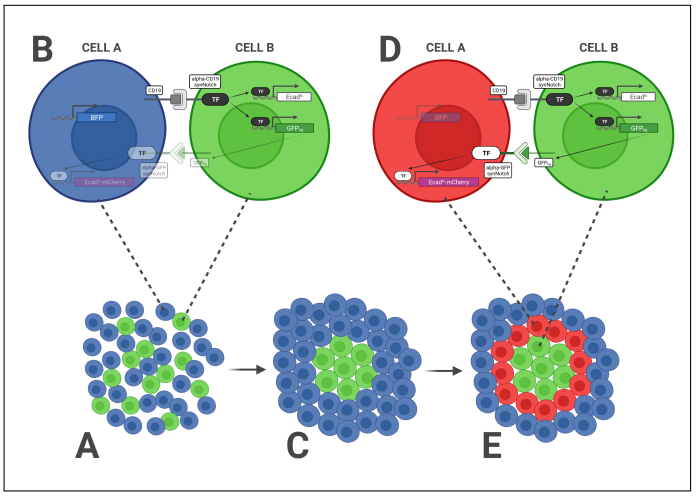


Fig. 9. The RGB synthetic morphogenesis process considered for the second case study. Adapted from [54].

9% of the corresponding compiled `nwn-snakes` model file, having 1807 lines of code (based on the files in the public `GitHub` repository, see Data Availability Statement).

Results recapitulate the emergence of the expected simplistic version of a synthetic morphogenetic pattern presented in [54]. Figure 10 depicts the evolving intensity of the fluorescent markers in each cell during a simulation of the `nwn-snakes` Python models compiled from the BiSDL description (see Alg. 4). At first (t = 10), the central cell slightly affects only one of its neighbors of the first degree, while other cells keep producing the BFP signal (Figure 10.A); the central cell engages all of its neighbors of the first degree, inducing the expression of mCherry in them, whose intensity evolves throughout the simulation (from t=20 to t=50, Figure 10.B-E); at t=60 all first-degree neighbors of the central cell express high levels of mCherry (Figure 10.F). On the contrary, the simulated deletion of GFP_{lig} results in a stable pattern with a central, colorless cell of type B and all its neighbors of the first and second degree constitutively expressing BFP (data not shown).

V. Conclusions

In conclusion, the BiSDL framework introduced here represents a significant advancement in the field of synthetic biology modeling. The core aim of BiSDL is to merge the detailed expressive capabilities of computational models with the user-friendliness of high-level languages, providing a tool that is both powerful and accessible. The hierarchical and modular structure of BiSDL is ideal for capturing the inherent complexity of biological systems, allowing for the construction of reusable and adaptable components.

Through the development of a prototype BiSDL compiler, models can now be compiled into simulations that encapsulate the spatial, hierarchical, and dynamic behaviors of biological entities, using the NWNs approach to handle biological complexity. Generated models rely on

```
MODULE rgb
    TIMESCALE 1
    SCOPE green_3_3 (3, 3)
        PROCESS GFP_production
            TIMESCALE 1
            TRANSCRIPTION (GFP_gene, GFP_mrna, INDUCERS:
              CD19_receptor_active_protein)
            TRANSLATION (GFP_mrna, 12*GFP_protein)
            DEGRADATION (GFP_protein)
            JUXTACRINE_SIGNAL (GFP_protein -> red_2_2)
            JUXTACRINE_SIGNAL (GFP_protein -> red_2_3)
            JUXTACRINE_SIGNAL (GFP_protein -> red_2_4)
            JUXTACRINE_SIGNAL (GFP_protein -> red_3_2)
            JUXTACRINE_SIGNAL (GFP_protein -> red_3_4)
            JUXTACRINE_SIGNAL (GFP_protein -> red_4_2)
            JUXTACRINE_SIGNAL (GFP_protein -> red_4_3)
            JUXTACRINE_SIGNAL (GFP_protein -> red_4_4)
    SCOPE grey_0_0 (0, 0)
        PROCESS CD19_production
            TIMESCALE 1 TRANSCRIPTION (CD19_gene,
              CD19_mrna)
            TRANSLATION (CD19_mrna, 2*CD19_protein)
            DEGRADATION (CD19_protein)
        PROCESS BFP_production
            TIMESCALE 1
            TRANSCRIPTION (BFP_gene, BFP_mrna)
            TRANSLATION (BFP_mrna, 2*BFP_protein)
            DEGRADATION (BFP_protein)
        PROCESS mCherry_production
            TIMESCALE 1
            TRANSCRIPTION (GFP_receptor_gene,
              GFP_receptor_mrna)
            TRANSLATION (GFP_receptor_mrna,
              GFP_receptor_protein)
            ENZYMATIC_REACTION
              (GFP_receptor_active_protein, [GFP_receptor_protein],
              [GFP_activation_mediator_protein])
            DEGRADATION (GFP_receptor_active_protein)
            TRANSCRIPTION (mCherry_gene, mCherry_mrna,
              INDUCERS: 2*GFP_activation_mediator_protein)
            TRANSLATION (mCherry_mrna, 3*mCherry_protein)
            DEGRADATION (mCherry_protein)
    SCOPE red_3_2 (3, 2)
        PROCESS CD19_production
        PROCESS BFP_production
        PROCESS mCherry_production
        JUXTACRINE_SIGNAL (CD19_protein -> green_3_3)
    .
    .
    .
    SCOPE red_4_4 (4, 4)
        PROCESS CD19_production
        PROCESS BFP_production
        PROCESS mCherry_production
        JUXTACRINE_SIGNAL (CD19_protein -> green_3_3)
    SCOPE blue_1_2 (1, 2)
        PROCESS CD19_production
        PROCESS BFP_production
        PROCESS mCherry_production
    .
    .
    .
    SCOPE blue_5_4 (5, 4)
        PROCESS CD19_production
        PROCESS BFP_production
        PROCESS mCherry_production
```

Algorithm 4: BiSDL description of the RGB synthetic morphogen system.

the `nwn-snakes` library, an extension of the `SNAKES` library [48] to support the NWNs formalism. Preliminary results indicate that BiSDL can dramatically simplify complex model descriptions, significantly reducing the code needed to represent sophisticated systems, as evidenced by the bacterial consortium and RGB case studies.

The accompanying `nwn-petrisim` simulator has been developed to execute and investigate behaviors of systems modeled in BiSDL, confirming that the BiSDL-compiled models are accurate representations of the expected behaviors of the systems studied.

Looking ahead, plans are in place to enhance BiSDL usability further by developing a visual language interface and refining the user interface to simplify the process of combining or editing modules. Efforts will also focus on integrating the parsing of Standard Biological Parts from the Registry into BiSDL modules and enabling BiSDL descriptions to be compiled into other modeling formalisms. A key area of future research will be to evaluate how

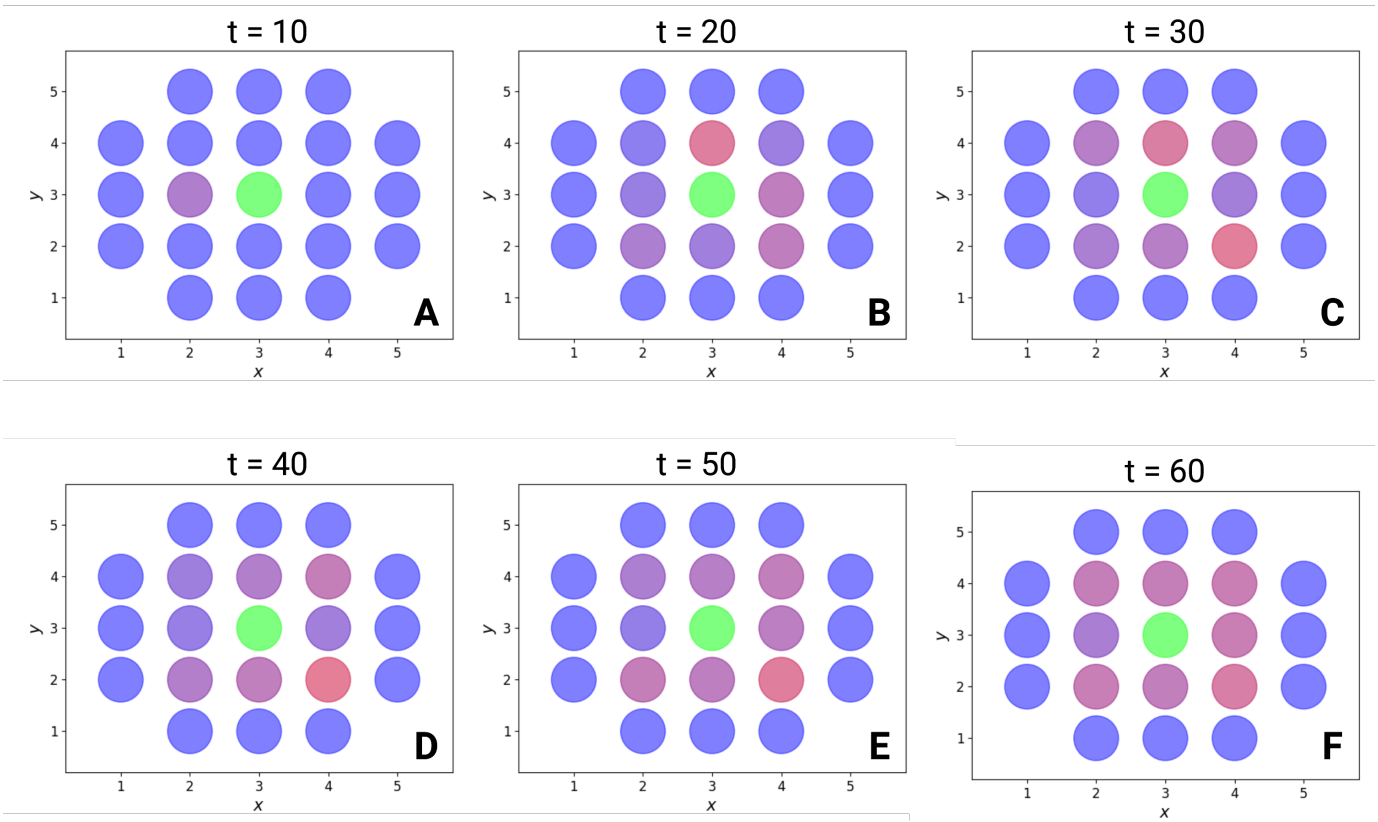


Fig. 10. Evolution of the three fluorescent marker levels (GFP, BFP and mCherry) in each cell on the two-dimensional spatial grid along the simulation.

BiSDL efficiency in reducing description length scales with increasing biological complexity. This work sets the stage for a more intuitive and efficient approach to modeling in synthetic biology.

Declarations

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Data and code availability

The code and data supporting this study and required to reproduce all published results are publicly available on GitHub and referenced with Zenodo at https://zenodo.org/doi/10.5281/zenodo.10478557.

Competing interests

The authors declare that they have no competing interests

Funding

This study was carried out within the project "SAI-SEI - Multi-Scale Protocols Generation for Intelligent Biofabrication" funded by the Ministero dell'Università e della Ricerca (Italian Ministry for Universities and Research) – within the Progetti di Rilevante Interesse Nazionale (PRIN) 2022 program (D.D.104 -02/02/2022) [Prot. 20222RT5LC].

Authors' contributions

LG: Conceptualization, Methodology, Software, Validation, Investigation, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization; **RB**: Conceptualization, Methodology, Software, Validation, Investigation, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization, Supervision, Funding acquisition; **AS**: Conceptualization, Methodology, Validation, Writing - Review & Editing, Supervision; **SDC**: Conceptualization, Methodology, Validation, Writing - Review & Editing, Supervision, Project administration, Funding acquisition.

Acknowledgements

Not applicable.

Authors' information

Leonardo Giannantoni is a Ph.D. candidate at the Department of Control and Computer Engineering of Politecnico di Torino. He holds an M.Sc. degree in Computer Engineering from Politecnico di Torino in Italy. His research focuses on the application of computational methods to synthetic biology.

Roberta Bardini is a post-doc researcher at the Department of Control and Computer Engineering of Politecnico di Torino. She holds a Ph.D. (2019) in Control and Computer Engineering from the Politecnico di Torino in Italy and an M.Sc. in Molecular Biotechnology (2014) from Università di Torino. Her research focuses on computational biology and bioinformatic approaches for analyzing and modeling complex biological systems and on biomanufacturing processes' computational design and optimization.

Alessandro Savino is an Associate Professor in the Department of Control and Computer Engineering at Politecnico di Torino (Italy). He holds a Ph.D. (2009) and an M.S. equivalent (2005) in Computer Engineering and Information Technology from Politecnico di Torino in Italy. Prof. Savino's research contributions include approximate computing, reliability analysis, safety-critical systems, software-based Self-tests, and image analysis. He has been part of the program and organizing committee of several IEEE and INSTICC conferences and has served as a reviewer of IEEE conferences and journals.

Stefano Di Carlo has been a full professor in the Control and Computer Engineering department at Politecnico di Torino (Italy) since 2021. He holds a Ph.D. (2003) and an M.Sc. equivalent (1999) in Computer Engineering and Information Technology from Politecnico di Torino in Italy. Di Carlo's research contributions include biological network analysis and simulation, machine learning, image processing, evolutionary algorithms, and development of resilient computer architectures.

References

[1] P. C. St. John and Y. J. Bomble, "Approaches to computational strain design in the multiomics era," *Frontiers in Microbiology*, vol. 10, p. 597, 2019.

[2] Y. Chen, D. Banerjee, A. Mukhopadhyay, and C. J. Petzold, "Systems and synthetic biology tools for advanced bioproduction hosts," *Current opinion in biotechnology*, vol. 64, pp. 101–109, 2020.

[3] N. S. McCarty and R. Ledesma-Amaro, "Synthetic biology tools to engineer microbial communities for biotechnology," *Trends in biotechnology*, 2018.

[4] G. Wu, Q. Yan, J. A. Jones, Y. J. Tang, S. S. Fong, and M. A. Koffas, "Metabolic burden: cornerstones in synthetic biology and metabolic engineering applications," *Trends in biotechnology*, vol. 34, no. 8, pp. 652–664, 2016.

[5] O. Borkowski, F. Ceroni, G.-B. Stan, and T. Ellis, "Overloaded and stressed: whole-cell considerations for bacterial synthetic biology," *Current opinion in microbiology*, vol. 33, pp. 123–130, 2016.

[6] M. R. Ebrahimkhani and M. Levin, "Synthetic living machines: A new window on life," *Iscience*, vol. 24, no. 5, 2021.

[7] L. Martini, G. Amprimo, S. Di Carlo, G. Olmo, C. Ferraris, A. Savino, and R. Bardini, "Neuronal spike shapes (nss): A straightforward approach to investigate heterogeneity in neuronal excitability states," *Computers in Biology and Medicine*, vol. 168, p. 107783, 2024.

[8] L. Martini, R. Bardini, A. Savino, and S. Di Carlo, "Gagam v1. 2: An improvement on peak labeling and genomic annotated gene activity matrix construction," *Genes*, vol. 14, no. 1, p. 115, 2022.

[9] R. Bardini, G. Politano, A. Benso, and S. D. Carlo, "Using multi-level petri nets models to simulate microbiota resistance to antibiotics," in *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Nov 2017, pp. 128–133.

[10] R. Bardini, S. Di Carlo, G. Politano, and A. Benso, "Modeling antibiotic resistance in the microbiota using multi-level petri nets," *BMC systems biology*, vol. 12, pp. 59–79, 2018.

[11] R. Bardini and S. Di Carlo, "Computational methods for biofabrication in tissue engineering and regenerative medicine-a literature review," *Computational and Structural Biotechnology Journal*, 2024.

[12] L. Giannantoni, R. Bardini, and S. Di Carlo, "A methodology for co-simulation-based optimization of biofabrication protocols," in *International Work-Conference on Bioinformatics and Biomedical Engineering*. Springer, 2022, pp. 179–192.

[13] L. Giannantoni, A. Savino, and S. Di Carlo, "Optimization of synthetic oscillatory biological networks through reinforcement learning," in *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. Istanbul: IEEE, 2023.

[14] R. Bardini, G. Politano, A. Benso, and S. Di Carlo, "Multi-level and hybrid modelling approaches for systems biology," *Computational and Structural Biotechnology Journal*, 2017.

[15] ——, "Computational tools for applying multi-level models to synthetic biology," in *Synthetic Biology*. Springer, 2018, pp. 95–112.

[16] B. Pouvreau, T. Vanhercke, and S. Singh, "From plant metabolic engineering to plant synthetic biology: The evolution of the design/build/test/learn cycle," *Plant Science*, vol. 273, pp. 3–12, 2018.

[17] F. Murphy, "Open access, open data, fair data and their implications for life sciences researchers," *Emerging Topics in Life Sciences*, vol. 2, no. 6, pp. 759–762, 2018.

[18] S. Stall, L. Yarmey, J. Cutcher-Gershenfeld, B. Hanson, K. Lehnert, B. Nosek, M. Parsons, E. Robinson, and L. Wyborn, "Make scientific data fair," 2019.

[19] S. M. Keating, D. Waltemath, M. König, F. Zhang, A. Dräger, C. Chaouiya, F. T. Bergmann, A. Finney, C. S. Gillespie, T. Helikar *et al.*, "Sbml level 3: an extensible format for the exchange and reuse of biological models," *Molecular systems biology*, vol. 16, no. 8, p. e9110, 2020.

[20] P. Gleeson, S. Crook, R. C. Cannon, M. L. Hines, G. O. Billings, M. Farinella, T. M. Morse, A. P. Davison, S. Ray, U. S. Bhalla *et al.*, "Neuroml: a language for describing data driven models of neurons and networks with a high degree of biological detail," *PLoS computational biology*, vol. 6, no. 6, p. e1000815, 2010.

[21] L. P. Smith, F. T. Bergmann, A. Garny, T. Helikar, J. Karr, D. Nickerson, H. Sauro, D. Waltemath, and M. König, "The simulation experiment description markup language (sed-ml): language specification for level 1 version 4," *Journal of integrative bioinformatics*, vol. 18, no. 3, p. 20210021, 2021.

[22] S. Konur, L. Mierla, H. Fellermann, C. Ladroue, B. Brown, A. Wipat, J. Twycross, B. P. Dun, S. Kalvala, M. Gheorghe *et al.*, "Toward full-stack in silico synthetic biology: integrating model specification, simulation, verification, and biological compilation," *ACS Synthetic Biology*, vol. 10, no. 8, pp. 1931–1945, 2021.

[23] E. Bartocci and P. Lió, "Computational modeling, formal analysis, and tools for systems biology," *PLoS computational biology*, vol. 12, no. 1, p. e1004591, 2016.

[24] R. Bardini, A. Benso, S. Di Carlo, G. Politano, and A. Savino, "Using nets-within-nets for modeling differentiating cells in the epigenetic landscape," in *International Conference on Bioinfor-*

matics and Biomedical Engineering. Springer, 2016, pp. 315–321.

[25] R. Bardini, A. Benso, G. Politano, and S. Di Carlo, "Nets-within-nets for modeling emergent patterns in ontogenetic processes," *Computational and Structural Biotechnology Journal*, vol. 19, pp. 5701–5721, 2021.

[26] F. Muggianu, A. Benso, R. Bardini, E. Hu, G. Politano, and S. Di Carlo, "Modeling biological complexity using biology system description language (bisdl)," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2018, pp. 713–717.

[27] R. Valk, "Object petri nets: Using the nets-within-nets paradigm, advanced course on petri nets 2003 (j. desel, w. reisig, g. rozenberg, eds.), 3098," *Appendix A: Proof of Theorem*, vol. 3, 2003.

[28] R. Bardini *et al.*, "A diversity-aware computational framework for systems biology," Ph.D. dissertation, PhD thesis. Politecnico di Torino, 2019 (cit. on p. 5), 2019.

[29] M. Hucka, D. P. Nickerson, G. D. Bader, F. T. Bergmann, J. Cooper, E. Demir, A. Garny, M. Golebiewski, C. J. Myers, F. Schreiber, D. Waltemath, and N. Le NovÃšre, "Promoting coordinated development of community-based information standards for modeling in biology: The combine initiative," *Frontiers in Bioengineering and Biotechnology*, vol. 3, p. 19, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fbioe.2015.00019

[30] COMBINE, "The COMBINE standards," [Online] https://co.mbine.org/standards, Aug. 2018.

[31] D. Waltemath, M. Golebiewski, M. L. Blinov, P. Gleeson, H. Hermjakob, M. Hucka, E. T. Inau, S. M. Keating, M. König, O. Krebs *et al.*, "The first 10 years of the international coordination network for standards in systems and synthetic biology (combine)," *Journal of integrative bioinformatics*, vol. 17, no. 2-3, 2020.

[32] "Combine specifications," http://co.mbine.org/standards/specifications/, 2021, accessed: 2021-01-29.

[33] R. N. Goldberg, M. Cary, and E. Demir, "Biopax a community standard for pathway data sharing," *Nature Biotechnology*, vol. 28, no. Nature Biotechnology, 2010.

[34] N. Le Novere, M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M. I. Aladjem, S. M. Wimalaratne *et al.*, "The systems biology graphical notation," *Nature biotechnology*, vol. 27, no. 8, p. 735, 2009.

[35] C. Chaouiya, D. Bérenguier, S. M. Keating, A. Naldi, M. P. Van Iersel, N. Rodriguez, A. Dräger, F. Büchel, T. Cokelaer, B. Kowal *et al.*, "Sbml qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools," *BMC systems biology*, vol. 7, no. 1, p. 135, 2013.

[36] C. M. Lloyd, M. D. Halstead, and P. F. Nielsen, "Cellml: its future, present and past," *Progress in Biophysics and Molecular Biology*, vol. 85, no. 2, pp. 433 – 450, 2004, modelling Cellular and Tissue Function. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S007961070400015X

[37] D. Köhn and N. Le Novère, "Sed-ml – an xml format for the implementation of the miase guidelines," in *Computational Methods in Systems Biology*, M. Heiner and A. M. Uhrmacher, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 176–190.

[38] M. Galdzicki, K. P. Clancy, E. Oberortner, M. Pocock, J. Y. Quinn, C. A. Rodriguez, N. Roehner, M. L. Wilson, L. Adam, J. C. Anderson *et al.*, "The synthetic biology open language (sbol) provides a community standard for communicating designs in synthetic biology," *Nature biotechnology*, vol. 32, no. 6, p. 545, 2014.

[39] R. S. Cox, C. Madsen, J. A. McLaughlin, T. Nguyen, N. Roehner, B. Bartley, J. Beal, M. Bissell, K. Choi, K. Clancy *et al.*, "Synthetic biology open language (sbol) version 2.2. 0," *Journal of integrative bioinformatics*, vol. 15, no. 1, 2018.

[40] C. Schölzel, V. Blesius, G. Ernst, and A. Dominik, "Characteristics of mathematical modeling languages that facilitate model reuse in systems biology: A software engineering perspective," *NPJ systems biology and applications*, vol. 7, no. 1, pp. 1–20, 2021.

[41] K. Choi, J. K. Medley, M. König, K. Stocking, L. Smith, S. Gu, and H. M. Sauro, "Tellurium: an extensible python-based modeling environment for systems and synthetic biology," *Biosystems*, vol. 171, pp. 74–79, 2018.

[42] M. Gutierrez, P. Gregorio-Godoy, G. Perez del Pulgar, L. E. Munoz, S. Sáez, and A. Rodríguez-Patón, "A new improved and extended version of the multicell bacterial simulator gro," *ACS synthetic biology*, vol. 6, no. 8, pp. 1496–1508, 2017.

[43] L. Bilitchenko, A. Liu, S. Cheung, E. Weeding, B. Xia, M. Leguia, J. C. Anderson, and D. Densmore, "Eugene–a domain specific language for specifying and constraining synthetic biological parts, devices, and systems," *PloS one*, vol. 6, no. 4, p. e18882, 2011.

[44] M. Pedersen and A. Phillips, "Towards programming languages for genetic engineering of living cells," *Journal of the Royal Society Interface*, vol. 6, no. suppl_4, pp. S437–S450, 2009.

[45] A. Basso-Blandin and F. Delaplace, "Gubs, a behaviour-based language for design in synthetic biology." *Scientific Annals of Computer Science*, vol. 23, no. 1, 2013.

[46] M. Galdzicki, C. Rodriguez, D. Chandran, H. M. Sauro, and J. H. Gennari, "Standard biological parts knowledgebase," *PloS one*, vol. 6, no. 2, p. e17005, 2011.

[47] B. Kienhuis, E. F. Deprettere, P. Van der Wolf, and K. Vissers, "A methodology to design programmable embedded systems," in *International Workshop on Embedded Computer Systems*. Springer, 2001, pp. 18–37.

[48] F. Pommereau, "Snakes: a flexible high-level petri nets library (tool paper)," in *International Conference on Applications and Theory of Petri Nets and Concurrency*. Springer, 2015, pp. 254–265.

[49] P. E. Purnick and R. Weiss, "The second wave of synthetic biology: from modules to systems," *Nature reviews Molecular cell biology*, vol. 10, no. 6, pp. 410–422, 2009.

[50] M. Amos and A. Goñi-Moreno, "Cellular computing and synthetic biology," in *Computational matter*. Springer, 2018, pp. 93–110.

[51] iGEM, "iGEM Parts Registry - Registry of Standard Biological Parts," 2023. [Online]. Available: http://parts.igem.org/

[52] A. L. Schaefer, D. L. Val, B. L. Hanzelka, J. E. Cronan, and E. P. Greenberg, "Generation of cell-to-cell signals in quorum sensing: acyl homoserine lactone synthase activity of a purified vibrio fischeri luxi protein," *Proceedings of the National Academy of Sciences*, vol. 93, no. 18, pp. 9505–9509, 1996.

[53] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz—open source graph drawing tools," in *Graph Drawing: 9th International Symposium, GD 2001 Vienna, Austria, September 23–26, 2001 Revised Papers 9*. Springer, 2002, pp. 483–484.

[54] S. Toda, L. R. Blauch, S. K. Y. Tang, L. Morsut, and W. A. Lim, "Programming self-organizing multicellular structures with synthetic cell-cell signaling," *Science*, vol. 361, no. 6398, pp. 156–162, 2018.

Appendix A
Basic building blocks

This section illustrates the syntax and semantics of the basic building blocks used to create a BiSDL description, focusing on accessibility and biological significance. Since BiSDL aims to be an easily readable language, it uses English keywords and minimal punctuation. It also avoids cumbersome solutions to delimit blocks. To this end, the use of indentations is suggested, although not mandatory, to set each block apart from the surrounding code visually.

The provided building blocks belong to different domains at Level II, where they provide descriptions of basic biological concepts, including simple local relations in the spatial domain, such as the relative position of two biological elements, and base functions in the behavioral domain, such as the production or degradation of a protein.

A. Process building blocks

This section provides a detailed overview of the basic constructs offered by BiSDL to model more complex behaviors inside `PROCESS` constructs. All parameters accepted by these building blocks are the base BiSDL types `gene`, `mrna`, `protein`, (protein) `complex`, and generic `molecule`, collectively indicated as *molecules* in the following. All molecule parameters can optionally be associated with a multiplier, i.e., an integer number indicating the ratio between the molecules involved in the process.

Listing 3. Transcription building block template

```
TRANSCRIPTION(<gene>, <mrna> (,
    regulation))
```

The construct in Listing 3 models the copying of DNA segments that can encode proteins in mRNA, the first step in gene expression, with the following parameters:

- `<gene>`: the gene to be transcribed; must end in "_gene";
- `<mrna>`: the transcribed mRNA; must end in "_mrna";
- (optional) `regulation` is a list of molecules acting as inhibitors, inducers, or activators (see subsection A-B).

Listing 4. Transcription building block example

```
TRANSCRIPTION(GFP_gene, GFP_mrna)
```

Listing 4 provides an example of the transcription of a gene to an mRNA.

Listing 5. Translation building block template

```
TRANSLATION(<mrna>, <protein> (,
    regulation))
```

The construct in Listing 5 models the process of protein synthesis from their mRNA blueprint, with the following parameters:

- `<mrna>`: the mRNA to be translated; must end in "_mrna";
- `<protein>`: the protein to be produced; must end in "_protein";
- (optional) `regulation` is a list of molecules acting as inhibitors, inducers, or activators (see subsection A-B).

Listing 6. Translation building block example

```
TRANSLATION(GFP_mrna, 2*GFP_protein)
```

Listing 6 provides an example of the translation of a mRNA to two proteins.

Listing 7. Degradation building block template

```
DEGRADATION(<molecule>)
```

The construct in Listing 7 models degradation of a molecule, with the following parameter:

- `<molecule>`: the molecule to be broken down.

Listing 8. Degradation building block example

```
DEGRADATION(3*GFP_protein)
```

Listing 8 provides an example of the degradation of a protein.

Listing 9. Complex formation building block template

```
PROTEIN_COMPLEX_FORMATION(<molecule>,
    <molecule>, ..., <molecule>)
```

The construct in Listing 9 models the process of two or more proteins associating in a protein complex, with the following parameter:

- `<molecule>, ...`: the proteins (at least two) that participate in the formation of the complex, followed by the name of the protein complex itself.

Listing 10. Complex formation building block example

```
PROTEIN_COMPLEX_FORMATION(3OC6HSL_molecule,
    LuxR_protein, 3OC6HSL_LuxR_complex)
```

Listing 10 provides an example of the formation of a protein complex.

Listing 11. Enzymatic reaction building block template

```
ENZYMATIC_REACTION(<protein>, [<molecule>,
    ...], [<molecule>, ...])
```

The construct in Listing 11 models the enzyme catalysis mechanism, through which a catalyst facilitates the reactants in the formation of one or more products, with the parameters:

- `<protein>`: the catalytic enzyme;
- `[<molecule>, ...]`: a list of at least one reactant, i.e., the *input* molecules;
- `[<molecule>, ...]`: the list of at least one product formed in the process, i.e., the output molecules.

Listing 12. Enzymatic reaction building block example

```
ENZYMATIC_REACTION(Lux1_protein,
    [SAM_molecule, ACP_molecule],
    [_3OC6HSL_molecule])
```

Listing 12 provides an example of an enzymatic reaction with two reagents and one product.

Listing 13. Custom process building block template

```
CUSTOM_PROCESS([<molecule>, ...],
    [<molecule>, ...] (, regulation))
```

The construct in Listing 13 allows the modeling of biological concepts not covered by the previously described ones, with the parameters:

- `[<molecule>, ...]`: a list of at least one reactant, i.e., the *input* molecules;
- `[<molecule>, ...]`: the list of at least one product formed in the process, i.e., the *output* molecules;
- (optional) `regulation`: a list of molecules acting as inhibitors, inducers, or activators (see subsection A-B).

Listing 14. Custom process building block example

```
CUSTOM_PROCESS([2*H2_molecule,
    O2_molecule], [2*H2O_molecule])
```

Listing 14 provides an example of a custom process.

B. Regulation

It is possible to optionally specify a list of regulatory mediators for the `TRANSCRIPTION`, `TRANSLATION`, and `CUSTOM_PROCESS` constructs, to increase or decrease the yield of the process.

Listing 15. Regulation mechanism template

```
<regulation\_type>: <molecule>, ... (,
    <regulation\_type>: ...)
```

Listing 15 shows the regulation construct, that accepts the following parameters:

- `<regulation_type>`: `INDUCERS`, `INHIBITORS`, or `ACTIVATORS`;
- `<molecule>, ...`: a list of molecules acting as inhibitors, inducers, or activators.

Listing 16. Regulation mechanism example

```
TRANSCRIPTION(GFP_gene, GFP_mrna,
    INDUCERS: 3OC6HSL_LuxR_complex)
```

Listing 16 exemplifies the transcription of a gene to an mRNA, induced by a protein complex.

C. Signaling mechanisms

The signaling mechanisms available in BiSDL enable the exchange of specific information between `SCOPE` entities:

- from a source entity towards its neighbors;
- from a source entity to a linked destination entity;
- bidirectionally, between two entities.

The first two – implemented by the constructs `PARACRINE_SIGNAL` and `JUXTACRINE_SIGNAL`, respectively – must be specified inside the `SCOPE` context, as they rely on the properties of the enclosing biological compartment. On the other hand, the specification of the third one (i.e., the `DIFFUSION` construct) must be inside the `MODULE` context after the description of all `SCOPE` statements, as it is a property equally shared between two `SCOPE` statements.

The BiSDL types that can be involved as `signal` parameters are the `protein`, `complex`, and `molecule` types (i.e., `gene`, `mrna`, and `receptor` types are not signals).

Listing 17. Paracrine signaling template

```
PARACRINE_SIGNAL  <signal> (, <signal>,
    ...)
```

Listing 17 models the communication mechanism from a cell to its neighboring cells. It requires the specification of a signal (or a list of signals). Spatial coordinates allow for automatically inferring the neighborhood of the container `SCOPE`. The signals reach the destination `SCOPE` unchanged. It accepts the following parameters:

- `<signal>, ...`: one or more `protein` or `molecule` entities acting as a signal;

Listing 18. Juxtacrine signaling template

```
JUXTACRINE_SIGNAL <signal> ->
    <entity_name>}
```

Listing 18 models contact-dependent signaling from a cell to a specific linked cell. It requires the signal and the receiving entity to be specified, with the following parameters:

- `<signal>`: the `protein` or `molecule` entity acting as a signal;
- `<entity_name>`: the destination `SCOPE`.

If the signal is a `molecule` (i.e., its name contains the tag "_molecule"), the corresponding construct models a cellular junction through which the signal is transferred from the source compartment to the destination one (Figure 11). On the other hand, if the signal involved is a `protein` (i.e., its name contains the tag "_protein"), it is interpreted as a membrane ligand and *consumed* in the bond with an active receptor (Figure 12 and Figure 21).

As a first example, the BiSDL module in Listing 19 models the mock process depicted in Figure 11 and is compiled to the PNs in Figure 20, where a communicating junction links two adjacent cells, allowing the molecule's transit, where it is modeled with the NWNs formalism as follows.

Listing 19. Modeling junction-mediated juxtacrine signaling

```
1 MODULE example
2     TIMESCALE 1
3     SCOPE s1 (0, 0)
```

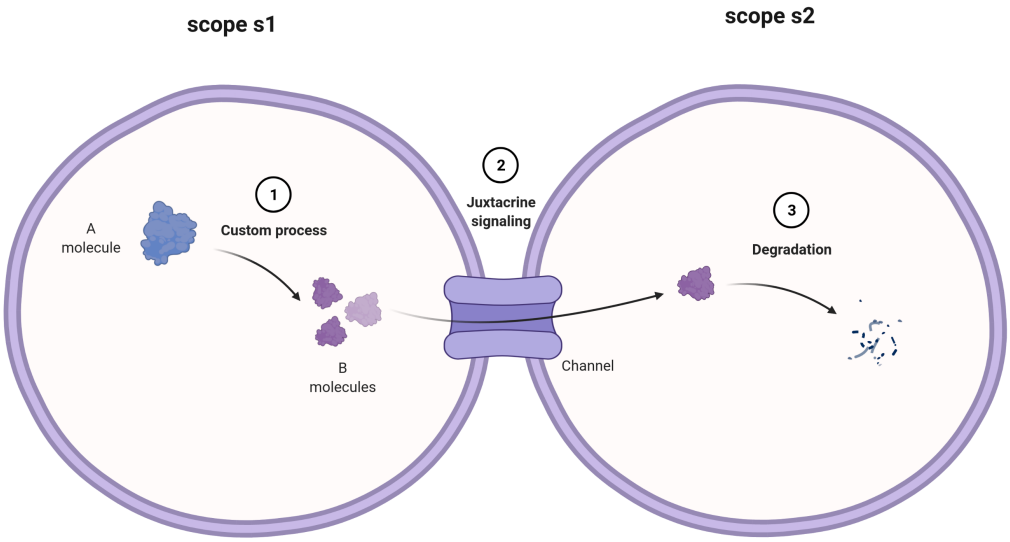


Fig. 11. This mock process modeled in Listing 19 involves two biological scopes, `s1` and `s2`. `s1` is the location where three `B_molecules` are produced (1) starting from one `B_molecule`. `B_molecules` are allowed to transit through a communicating junction from scope s1 to scope s2 (2). In `s2`, `B_molecules` are constantly degraded (3).

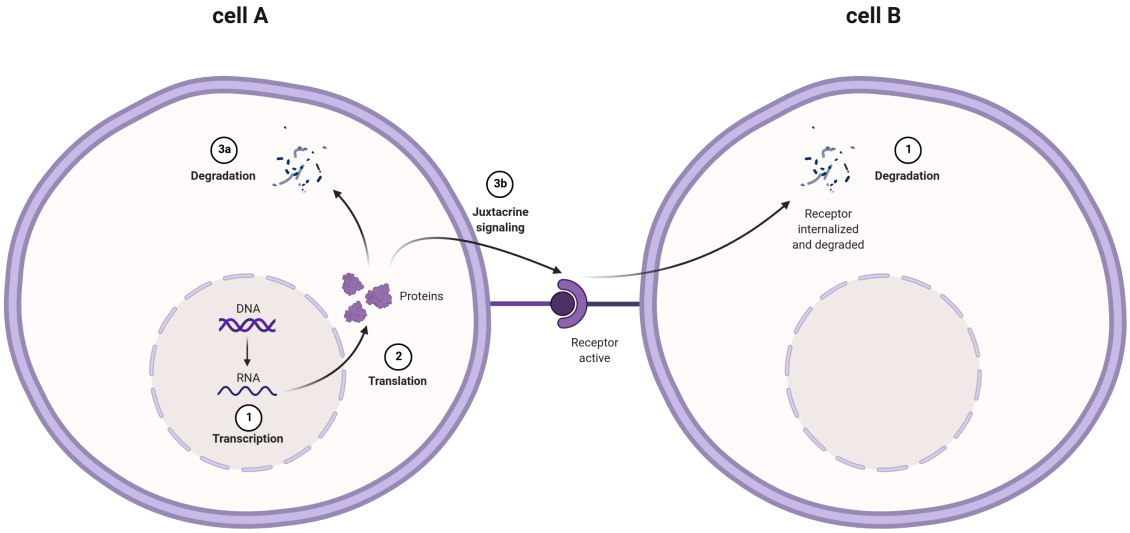


Fig. 12. This mock process modeled in Listing 20 involves two biological scopes, `cell_A` and `cell_B`. `cell_A` is the location where three `A_molecules` are produced (2) starting from gene-mRNA transcription (1). `A_molecules` may face degradation (3a), or they could bind (3b) the corresponding receptor on `cell_B` membrane. The active receptor is subject to degradation (`cell_B`, (4)).

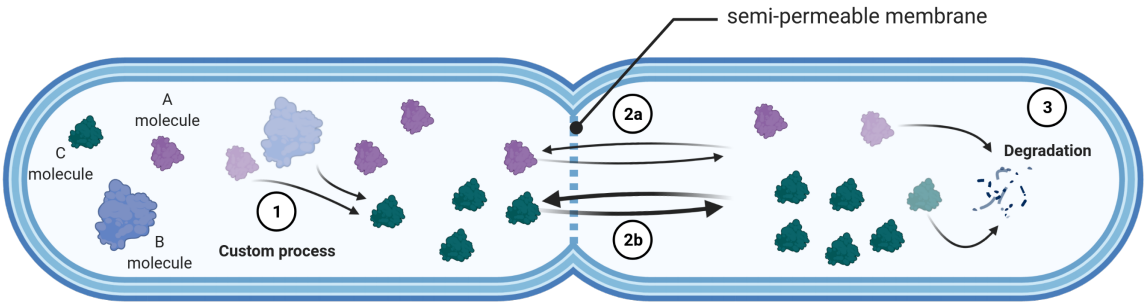


Fig. 13. This mock process modeled in Listing 22 involves two biological compartments separated by a semi-permeable membrane allowing the diffusion of `A_molecule` and `C_molecule`. The latter diffuses (2b) twice *as easily* as the former (2a) through the membrane. The compartment left to the membrane is the location where `C_molecule` is produced (1) starting from one `A_molecule` and one `B_molecule`. In the rightmost compartment, the only transformation is the degradation (3) of molecules A and B.

```
4           PROCESS p1
5               TIMESCALE 1
6               CUSTOM_PROCESS ([A_molecule],
                    [3*B_molecule])
7           JUXTACRINE_SIGNAL B_molecule -> s2
8       SCOPE s2 (0, 1)
9           PROCESS p2
10              TIMESCALE 1
11              DEGRADATION (B_molecule)
```

As a second example of juxtacrine signaling, Listing 20 models the biological system in Figure 12, where two adjacent cells interact through a membrane ligand (`A_protein`) and a membrane receptor (`A_receptor_active_protein`).

Listing 20. Modeling ligand-receptor juxtacrine signaling

```
1  MODULE example
2      TIMESCALE 1
3      SCOPE cell_A (0, 0)
4          PROCESS process_A
5              TIMESCALE 1
6              TRANSCRIPTION (A_gene, A_mrna)
7              TRANSLATION (A_mrna,
                   3*A_protein)
8              DEGRADATION (A_protein)
9          JUXTACRINE_SIGNAL A_protein ->
               cell_B
10     SCOPE cell_B (0, 1)
11         PROCESS process_B
12             TIMESCALE 1
13             DEGRADATION
                   (A_receptor_active_protein)
```

In BiSDL, the declaration of the juxtacrine signaling from `cell_A` to `cell_B` (line 9, Listing 20) using the `A_protein` ligand builds the whole ligand-receptor underlying structure. That is, `SCOPE cell_B` is automatically endowed with the (virtual) permanent presence of a membrane receptor for `A_protein` ligand. When the `juxtacrine_signaling_A_molecule_s1_s2_1` transition (Figure 21.a) fires, one `A_protein` token is consumed in `cell_A` and one `A_receptor_active_protein` token is produced in `cell_B`, with the text substitution signifying the activation of the bond. Therefore, there is no need for additional modeling of the signaling in the description of `SCOPE cell_B` behavior (lines 10-13, Listing 20). The `DEGRADATION` process (line 13, Listing 20) models the degradation of ligand-receptor complexes.

Listing 21. Diffusion template

```
DIFFUSION  <entity_name>, <entity_name>,
    <signal> (, <signal>, ...)
```

Listing 21 models a bi-directional permeable membrane between two biological districts, allowing the movement of signals between them, with the following parameters:

- `<entity_name>, <entity_name>`: the two communicating `SCOPE` entities;
- `<signal>, ...`: one or more `signal` entities the virtual membrane is permeable to.

Listing 22. Modeling diffusion

```
1  MODULE example
2      TIMESCALE 1
3      SCOPE s (0, 0)
4          PROCESS ps
5              TIMESCALE 1
6              CUSTOM_PROCESS ([A_molecule,
                   B_molecule], [C_molecule])
7      SCOPE t (0, 1)
8          PROCESS pt
9              TIMESCALE 1
10             DEGRADATION (A_molecule)
11             DEGRADATION (C_molecule)
12     DIFFUSION s, t, [A_molecule,
           2*C_molecule]
```

Listing 22 is the BiSDL representation of the biological phenomena in Figure 13. A biological district is divided in two by a membrane. In the leftmost part (a) in Figure 13, a reaction occurs through which `C_molecule` is produced starting from `A_molecule` and `B_molecule`. The membrane is permeable to molecules A and C, with a higher permeability for C. This different permeability is modeled in BiSDL using a multiplier (line 12, Alg. 22).

Appendix B
From BiSDL to NWNs

This section details the mapping between BiSDL constructs and the chosen NWNs target representation through the `nwn-snakes` library. The `Module` class is a utility layer that eases the simulation task, automatically wrapping the NWNs model structure compiled from the BiSDL model description. To facilitate the declaration of variables, the specification of NWNs elements in the `Module` class follows the order: PNs, `places`, `Transitions`, and finally input and output arcs involving the preceding elements of any level. Each BiSDL `PROCESS` (see Section A-A), once compiled, maps to a net token where places and transitions deal only with black tokens. The black tokens at the lower level correspond, in the top-level *container* place, to colored tokens, i.e., strings bearing the name of the lower-level place holding them. places that are shared among constructs of the same `PROCESS` context are instantiated only once, while their interactions (in terms of arcs and transitions) are created according to the modeled processes.

A. Compiled process building blocks

As an example, The BiSDL `TRANSLATION` construct in Listing 6 compiles into the PNs shown in Figure 14, having one `GFP_mrna` place and one `GFP_protein` place. The two are connected by one `GFP_mrna_translation` transition that, once fired, consumes one token from the `GFP_mrna` place and produces two in the `GFP_protein` place. In this example process, where two proteins (the `(dot, dot)` black tokens) are produced when one mRNA black token (`dot`) is consumed.

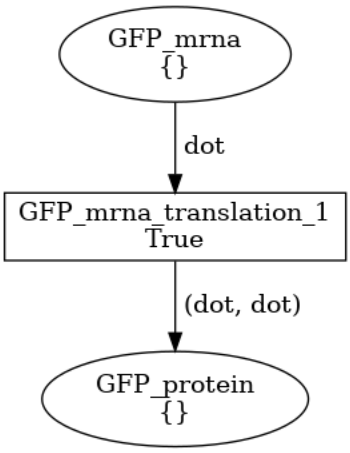


Fig. 14. PNs for the translation process, generated from the BiSDL code for `TRANSLATION` in Listing 6. Two proteins (the `dot, dot` black tokens) are produced when one mRNA black token is consumed.

As another example, the BiSDL `DEGRADATION` construct in Listing 8 compiles into the PNs shown in Figure 15: a `GFP_protein` place is created and connected to three separate transitions that can fire independently, thus depleting up to three tokens.

For the `ENZYMATIC_REACTION` construct, Listing 12 compiles into the PNs shown in Figure 16. In this example, `Lux1_protein` catalyzes the reaction process involving the reactants `SAM_molecule` and `ACP_molecule`, leading to the formation of `3OC6HSL_molecule`. A place is created for the catalytic enzyme and one for each reactant and product. The reactants are input places for the transition representing the enzymatic reaction, which produces one token in the `3OC6HSL_molecule` place upon firing. At the same time, one token is produced back in the `Lux1_protein` place, which, being a catalyst, is not consumed nor changed by the reaction.

The BiSDL `CUSTOM_PROCESS` construct in Listing 14 compiles into the PNs shown in Figure 17: `H2_molecule` and `O2_molecule` participate in the formation of `H2O_molecule`. One place is created for each molecule. When the transition fires, two black tokens are produced in the `H2O_molecule` place, while two black tokens are depleted from the `H2_molecule` and one from the `O2_molecule`.

In general, BiSDL descriptions holding these constructs generate a two-level hierarchy of nets. For example, in the context of a complete BiSDL description, the `CUSTOM_PROCESS` construct in Listing 14 generates the described PNs model (see Figure 17, and 18.b) at the lower level, and a single place holding it at the top level (Figure 18.a).

B. Compiled regulation constructs

To obtain regulation of basic building block operation, inducer molecules are mapped to additional input places to their main transition. As an example, The BiSDL `TRANSCRIPTION` construct in Listing 16 compiles into the PNs shown in Figure 19, including a `GFP_gene` place, a `GFP_mrna` place, and a `3OC6HSL_LuxR_complex` place, connected by one transition. The `GFP_gene_transcription` transition has two input arcs from the `GFP_gene` place and the `3OC6HSL_LuxR_complex` place and two output arcs towards the `GFP_gene` and the `GFP_mrna` place. The gene place is automatically initialized with one black token. The gene token is consumed when the transition fires and is immediately created back. Thus, the net result upon firing is the production of one token in the `GFP_mrna` place and the expenditure of one token in `3OC6HSL_LuxR_complex` place.

On the contrary, inhibitory regulation is modeled, at the PNs level, with an additional transition competing for the tokens in input to the main transition.

C. Compiled signaling constructs

The PNs structures produced upon compilation of signaling constructs are transitions and arcs that link places in the top-level net (i.e., BiSDL `SCOPE`s).

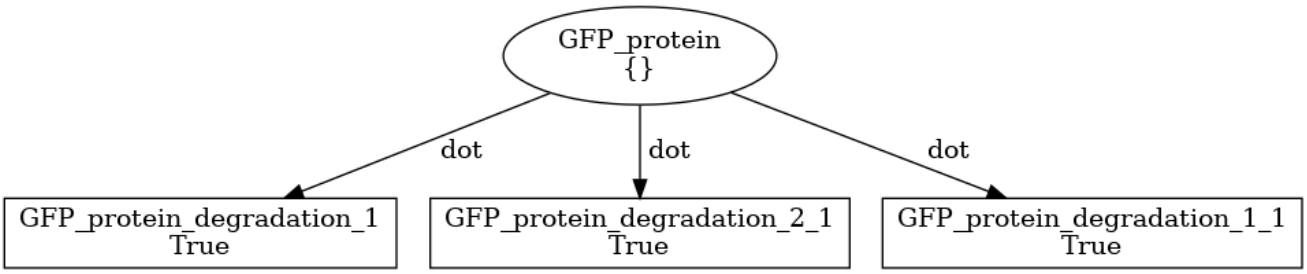


Fig. 15. PNs for the degradation process corresponding to the construct for `DEGRADATION` in Listing 8. When each one of the transitions fires, one protein is consumed.

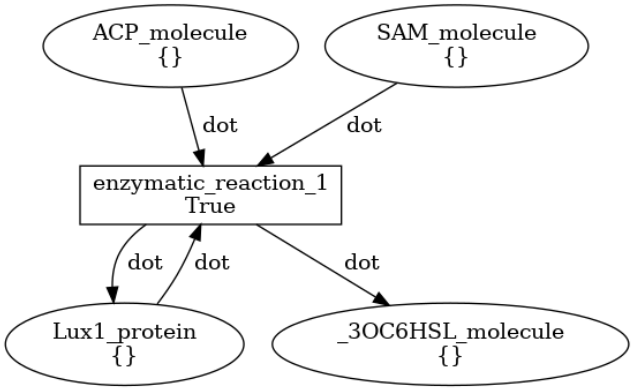


Fig. 16. PNs for the enzymatic reaction process corresponding to the construct for `ENZYMATIC_REACTION` in Listing 12. When the transition fires, one black token is consumed from each of the input places, and one token is produced in the output place (`3OC6HSL_molecule`) and in the place representing the catalysts (`Lux1_protein`).

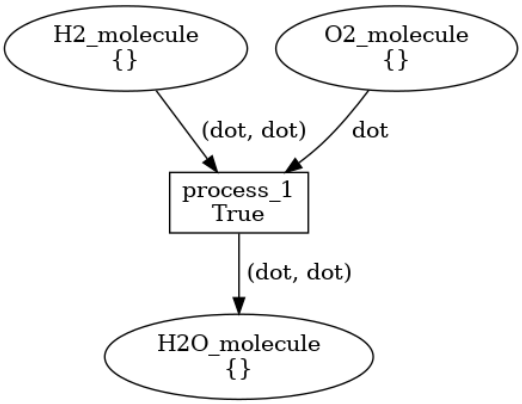


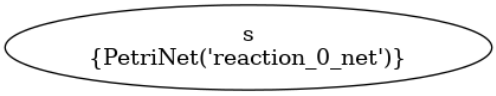
Fig. 17. PNs for the custom process corresponding to the construct for `CUSTOM_PROCESS` in Section Listing 14. When the transition starts, two black tokens are consumed from the `H2_molecule` input places, one from `O2_molecule` input places, and two black tokens are produced in the output place (`H2O_molecule`).

For example, Listing 19 compiles into the NWNs model in Figure 20.

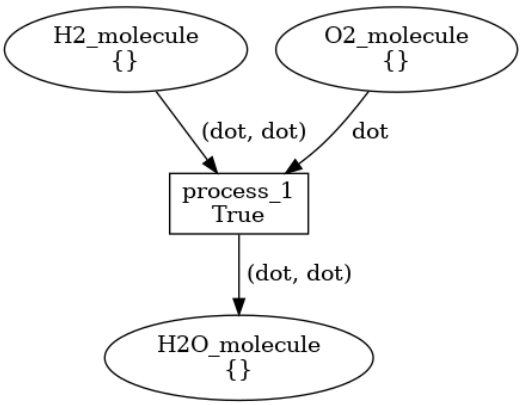
In Figure 20.a, `p1_net` is a net token in place `s1`, and `p2_net` is a net token in place `s2`. Their internal machinery is represented in Figure 20.b and Figure 20.c, respectively. `p1_net` and `p2_net` evolution takes place entirely in their respective enclosing places: net tokens do not move across top-level places. Figure 20.b shows the PNs `p1_net` corresponding to process p1 (lines 4-6, Listing 19). This is the internal machinery of the colored (net) token inside `place s1` in Figure 20.a. The `process_1` transition gets black tokens from the input place `A_molecule` and produces black tokens in the output place `B_molecule`. Finally, Figure 20.c shows the PNs `p2_net`, corresponding to process p2 (lines 9-11, Listing 19), and to the internal machinery of the colored (net) token inside `place s2` in Figure 20.a. The `B_molecule_degradation` transition depletes black tokens from the input place `B_molecule`. Black tokens produced and consumed in `p1_net` places (Figure 20.b) correspond to colored tokens (strings) in the top level `s1` place (Figure 20.a). In general, a low-level black token corresponds to a top-level colored token by the same name of the place it is produced or consumed in. The transition `juxtacrine_signaling_A_molecule_s1_s2_1` is enabled by `A_molecule` colored tokens from `s1`, and outputs the same colored tokens in the output place `s2`. Note that the substitution rule `"protein", "receptor_active_protein"` has no effect in this example, thus letting `A_molecule` tokens move to the destination scope unchanged. In fact, the biological process we are modeling lets the scopes communicate through a junction (the *Channel* in Figure 11). The scope of this substitution is to model the activation of a receptor when the `JUXTACRINE_SIGNAL` construct is used to model the juxtacrine interaction between a ligand and a membrane receptor, as illustrated in the next example.

The PNs structure built after a `DIFFUSION` construct consists, in general, of two transitions per signal. One transition consumes tokens from the first `<entity_name>` and produces the same amount of tokens in the second

<entity_name>. The other transition works in the opposite direction. When multipliers are specified for a signal, the PNs structure produced will be composed of as many pairs of transitions as specified by the multiplier.



(a) Top-level PNs. place s represents the biological district where the reaction occurs. The reaction is secluded in the colored (net) token PetriNet('reaction_0_net').



(b) Bottom-level PNs. This is the internal machinery of the colored (net) token inside place s (left), where the reaction at line 6, Listing 14, occurs.

Fig. 18. Graphical representation of the PNs generated by the compiled BiSDL example in Listing 14.

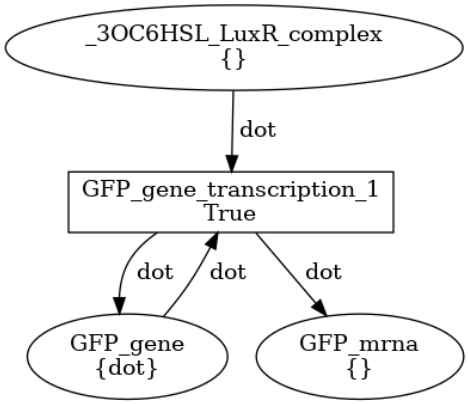
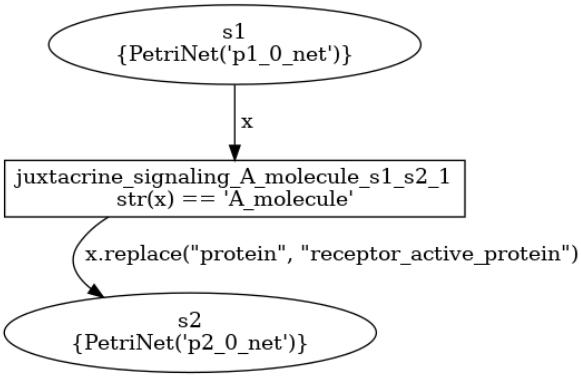
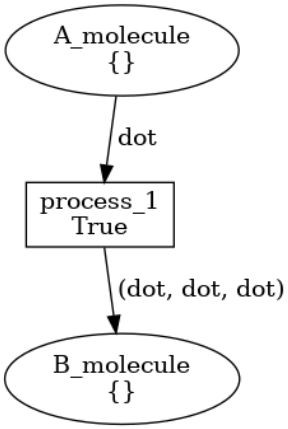


Fig. 19. PNs for the transcription process, generated from Listing 4, the BiSDL code for TRANSCRIPTION in Section A-A.

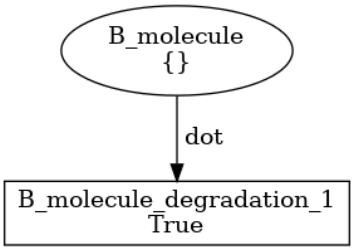
As an example, Listing 22 compiles into NWNs structure in Figure 22. Figure 22.a shows the top-level net structure. places s and t are mutually connected by one couple of diffusion_A_molecule_<N> transitions allowing "molecule_A" colored tokens bi-directional passage, and by two couples of diffusion_C_molecule_<M> transitions for "molecule_C" tokens passage. Such structure allows C_molecule to traverse the membrane with twice the probability with respect to A_molecule.



(a) Top-level PNs. p1_net is a net token in place s1 and p2_net is a net token in place s2. Their internal machinery is represented in (b) and (c), respectively.

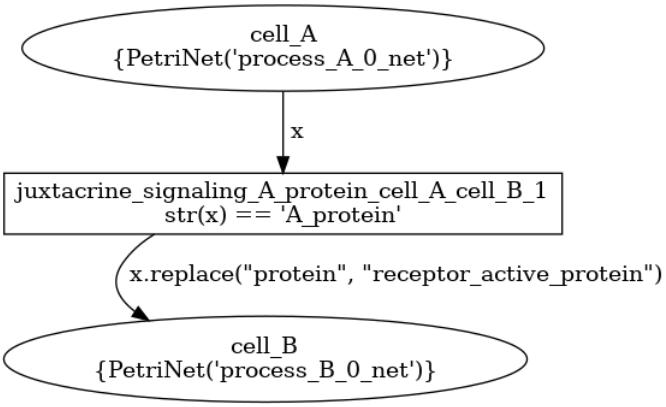


(b) Bottom-level PNs p1_net corresponding to the net token in place s1 (a).

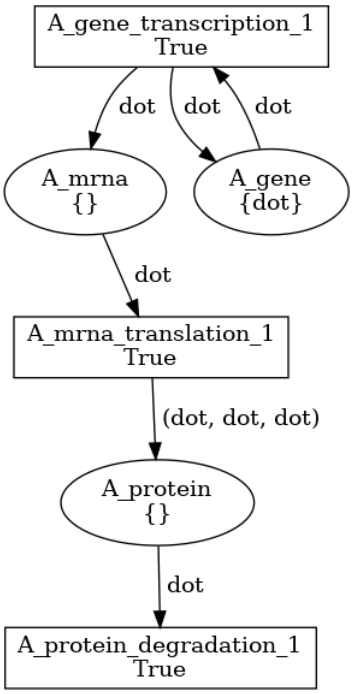


(c) Bottom-level PNs p2_net corresponding to the net token in place s2 (a).

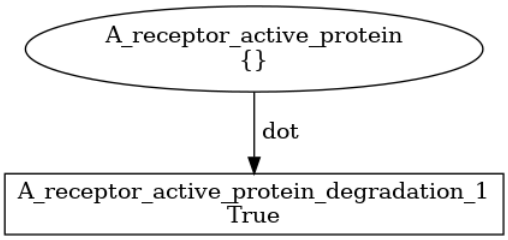
Fig. 20. Graphical representation of the PNs generated by the compiled BiSDL example in Listing 19.



(a) Top-level PNs. `process_A_net` is a net token in place `cell_A` and `process_B_net` is a net token in place `cell_B`. Their internal machinery is represented in (b) and (c), respectively.

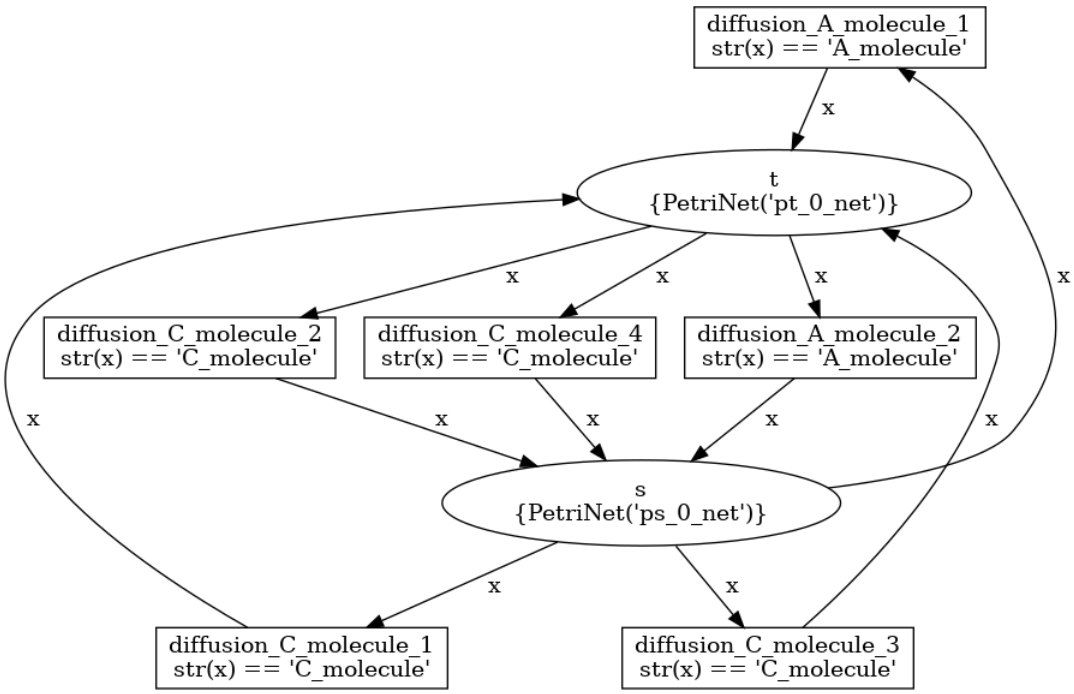


(b) Bottom-level PNs `process_A_net` corresponding to the net token in place `cell_A` (a).

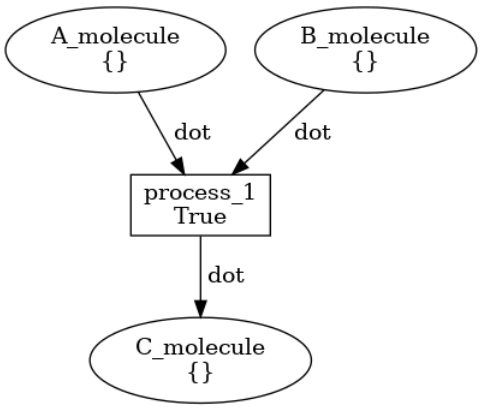


(c) Bottom-level PNs `process_B_net` corresponding to the net token in place `cell_B` (a).

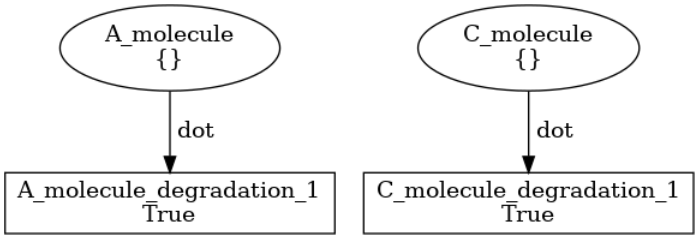
Fig. 21. Graphical representation of the PNs generated by the compiled BiSDL example in Listing 20.



(a) Top-level PNs. `ps_0_net` is a net token in place s and `pt_0_net` is a net token in place t. Their internal machinery is represented in (b) and (c), respectively.



(b) Bottom-level PNs `ps_0_net` corresponding to the net token in place s (a).



(c) Bottom-level PNs `pt_0_net` corresponding to the net token in place t (a).

Fig. 22. Graphical representation of the PNs generated by the compiled BiSDL example in Listing 22.