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

BMC Bioinformatics

RESEARCH

Open Access

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

Leonardo Giannantoni[1], Roberta Bardini[1*], Alessandro Savino[1] and Stefano Di Carlo[1]

*Correspondence:
roberta.bardini@polito.it

[1] Department of Control and Computer Engineering, Polytechnic University of Turin, Corso Duca degli Abruzzi, 24, 10129 Turin, TO, Italy

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 three case studies on complex multicellular systems: a bacterial consortium, a synthetic morphogen system and a conjugative plasmid transfer process. 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 designing and simulating 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.

Keywords: Systems biology, Synthetic biology, Computational biology, Domain-specific languages

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 [11–13] of complex biological processes. In particular, for analyzing and predicting the dynamics of multicellular synthetic systems, 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 Sect. "Related work"). 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 [23–25] 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 (NWN) 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.

BiSDL closely mirrors the natural language used within the biological domain. The compiler manages the gap between this high-level biological semantics and the low-level NWN formalism syntax, reducing the need for advanced modeling skills and knowledge of the low-level formalism. While in its current implementation BiSDL requires basic programming and modeling skills, the language could be integrated with a dedicated Graphical User Interface (GUI), paving the way for extensive broadening of the user base. BiSDL aims to simplify data exchange in bioinformatics, offering a high-level approach that abstracts away the complexity found in standards like SBOL [29, 30] and SBML [31], and is versatile enough to be translated into other formalisms like Petri Nets (PN), unlike tools such as pySBOL [32] and libSBML [33] that still rely on complex XML-like syntax.

The paper is organized as follows: Sect. "Related work" summarizes existing computational languages for synthetic biology, Sect. "Methods" details the design, syntax, and semantics of BiSDL and its compilation into low-level models using the NWN formalism. Then, Sect. "Results and discussion" showcases BiSDL capabilities through three case studies on multicellular synthetic systems. Finally, Sect. "Conclusions" summarizes the contributions, highlights open challenges, and outlines future developments.

Related work

The scientific landscape of model description languages for systems and synthetic biology is rich and complex.

Figure 1 organizes them by expressivity of biological semantics and generality in modeling different biological levels or domains. Each language links to the biological levels it targets (either molecular pathways, cells, multicellular systems, or a combination thereof) and the level of flexibility the language has in generalizing to different biological domains or mechanisms (see Legend on the right).

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 [34–36]. 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 Equations (ODE) models, making this mathematical modeling formalism accessible by non-expert users. Some of the COMBINE DSLs specialize in intracellular pathways, such as BioPAX [37], and processes, such as Systems Biology Graphical Notation (SBGN) [38], SBML [31] and CellML [39]. SED-ML [21, 40] 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 [29, 30] targets *in silico* synthetic genetic designs, yet is limited to genetic circuits alone, and does not cover any other biological aspect. Existing standards such as SBOL [29, 30], CellML [39], SED-ML [21, 40], and SBML [31] offer valuable frameworks for data exchange, albeit

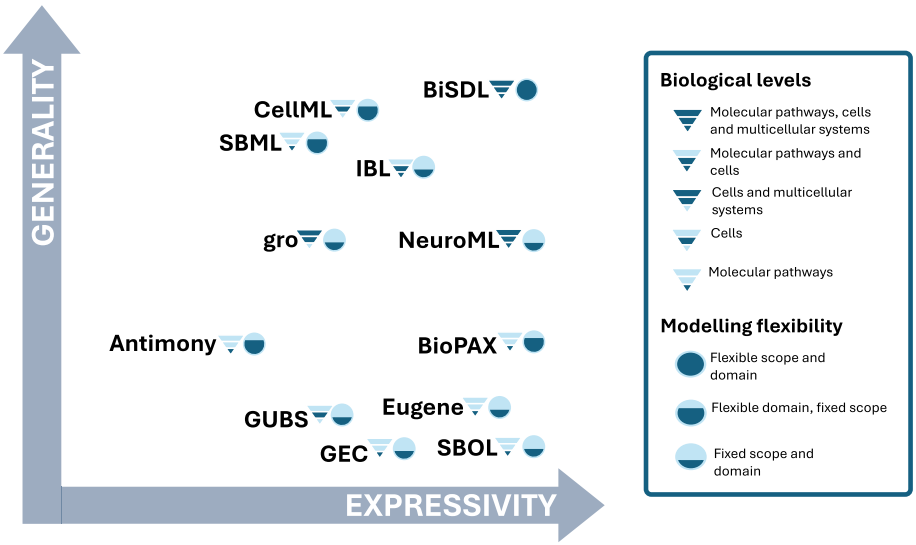

Fig. 1 Comparison of Model Description Languages in Systems Biology over expressivity (broadness and depth of described models) and generality (broadness of modeling target, scope, and domain), providing further details on the biological levels covered and the modeling flexibility supported

with limitations such as verbosity and complexity. While tools like pySBOL [32] and libSBML [33] provide programmatic access to these standards, they still require users to navigate XML-like syntax. BiSDL allows users to focus solely on high-level concepts, abstracting from the implementation details. The proposed compiler, translating BiSDL into PN, showcases the versatility of the language, demonstrating the potential for translation into other languages and formalisms.

Besides COMBINE standards, several computational languages for systems and synthetic biology exist [41]. 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 [42]. The Cell Programming Language (gro) [43] 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 [44] specifies synthetic biological parts, devices, and systems, mainly focusing on genetic constructs and their expression. Genetic Engineering of Cells (GEC) [45] centers over logical interactions between proteins and genes. GEC programs can be compiled into sequences of standard biological parts for laboratory applications. Genomic Unified Behavior Specification (GUBS) [46] 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, both languages require programming skills. Thus, neither 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 [47]. 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 nor 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.

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-friendly biological semantics with the capacity to capture system complexity. Its development is centered on domain-specific terminology and the ability to compile into NWN simulation models, discussed in Sect. 3.4. 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 NWN models for the simulation and analysis of system behavior.

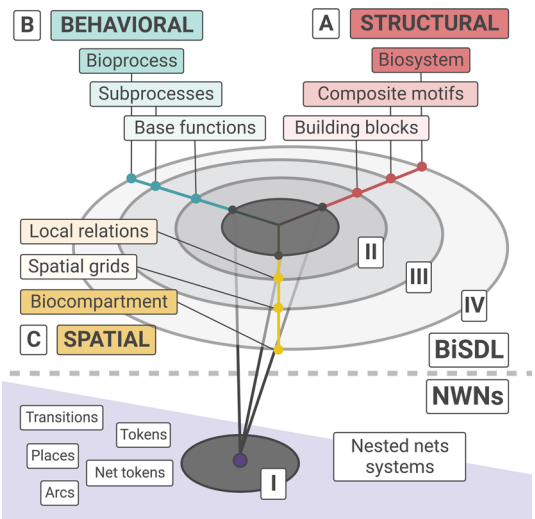

Fig. 2 A scheme of BiSDL domains and abstraction levels inspired to the VHDL Y-Chart [48]. The upper panel (BiSDL) shows the (A) Structural, (B) Behavioral, and (C) Spatial domains and the corresponding high abstraction levels (II, III, and IV) in BiSDL descriptions. The lower panel (NWN) illustrates the three domains at the low abstraction level relative to the Nets-Within-Nets formalism (I)

Biological perspectives and levels of abstraction

The BiSDL syntax supports describing spatial and multi-level biological concepts through multiple domains and abstraction levels, as illustrated in the Y Chart reported in Fig. 2.

Inspired by the Very High-Speed Integrated Circuit Hardware Description Lan-guage (VHDL) [48], 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 Fig. 2 (A—STRUCTURAL, top right), 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 (Fig. 2, B - BEHAVIORAL, top left) 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 (Fig. 2 C—SPATIAL, center left) 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 NWN modeling formalism elements (low abstraction).

Level IV (Fig. 2, circle 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* (Fig. 2, circle 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* (Fig. 2, circle 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* (Fig. 2, circle I) describes NWN formalism elements combined in low-level models of the system.

When describing composite systems (e.g., a biological tissue), BiSDL covers all domains: structural, spatial, and behavioral (as shown in Fig. 2.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 a set of composable BiSDL descriptions are detailed in Additional file 1 BiSDL Modules Library—Section 1 as an example of a BiSDL library.

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 compartments 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). Integer timescales can represent any ratio between the operations of different models in the provided discrete-time simulator. The TIMESCALE of a module (line 2) sets the base pace of the system dynamics compared to the unitary step of the discrete-time simulator. For instance, if one model has a TIMESCALE of N, it means that it evolves by 1 step every N simulator's steps (whose TIMESCALE is made equal to 1). The model is slower than the base time step by a factor of N.

Each SCOPE contains a set of biological PROCESS instantiations with explicit identifiers (lines 4–8 and 9). They 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. The same applies to processes with different timescales: they proceed at a relative speed, the ratio of their respective timescales. For instance, if TIMESCALE is 2 for PROCESS *p1*, and 5 for PROCESS *p2*, they will proceed at a relative speed of 5/2 (i.e., *p1* evolves 2.5 times faster than *p2*). Different PROCESS instances can connect over the same elements: for example, one process might produce a molecule that regulates a base function in another process. BiSDL emphasizes ease of description. Each SCOPE can reuse a PROCESS from another SCOPE simply by declaring a PROCESS with the same <process_id>.

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.

```
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>, ...]
       ⋮
```

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.

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.

```
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])
```

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 molecular 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.

BiSDL constructs

This work proposes a library of BiSDL constructs (see Additional file 1—Section 1) to exemplify the language semantic capabilities, showing its expressiveness and closeness to the biological semantics. To foster standardization in model description languages, all proposed BiSDL constructs follow the Systems Biology Ontology (SBO)

[49] and fall into the following four subcategories (of the seven provided by the standard):

- Physical entity representation:
  - Material entities identify the functional entities (i.e., `SCOPE`, `CELL`, and the base types `GENE`, `MRNA`, `PROTEIN`, `COMPLEX`, `MOLECULE`);
  - Functional entities identify the function they perform (i.e., `PARACRINE_SIGNAL`, `JUXTACRINE_SIGNAL`, and `DIFFUSION`).
- Participant role:
  - identifies the role played by an entity in a modeled process (i.e., `INDUCERS`, `INHIBITORS`, `ACTIVATORS`);
- Occurring entity representation:
  - identifies processual relationships involving physical entities (i.e., `TRANSCRIPTION`, `TRANSLATION`, `DEGRADATION`,`PROTEIN_COMPLEX_FORMATION`, `ENZYMATIC_REACTION`, `CUSTOM_PROCESS`);
- System description parameter:
  - provides quantitative descriptions of biological processes (i.e., `TIMESCALE`s and the multipliers of physical entities).

From BiSDL descriptions to NWNs models

BiSDL supports the system dynamics analysis utilizing simulations. This is obtained by compiling BiSDL descriptions into low-level models based on the NWN formalism.

Compilation of NWNs models

NWN extend the PN formalism to support hierarchy, encapsulation, and selective communication [9, 24, 27], which makes them suitable to model complex biological processes [25, 28], coherently with the design goals of BiSDL (Sect. 3.1). PN are bipartite graphs where nodes 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 NWN formalism is a high-level PN formalism supporting all features of other high-level PN: tokens of different types and timed and stochastic time delays associated with transitions. NWN 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, NWN support hierarchical organization, and each layer relies on the same formalism (see Fig. 3). This characteristic introduces the Object-Oriented Programming (OOP) paradigm within the PN formalism. Therefore, NWN models express encapsulation and

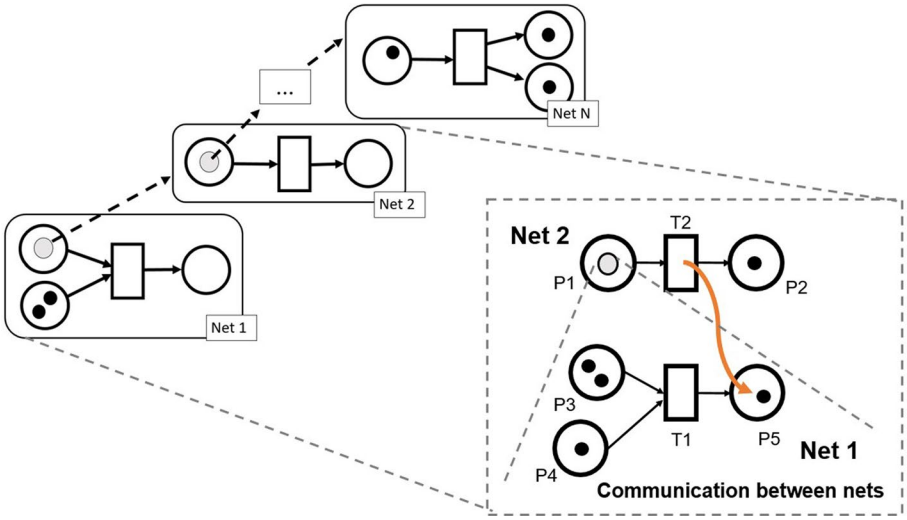

Fig. 3 A representation of the NWN formalism. Tokens in these PN can be instances of PN, thus implementing a hierarchy of encapsulated levels. Channels can interlock two transitions from different nets, allowing the exchange of tokens and information

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.

NWN have heightened expressivity compared to other modeling formalisms. While Boolean models offer binary node states, high-level Petri Nets can convey intricate information regarding system resources and processes. Similarly, while ODE represent uniform compartments with continuous values, NWN accommodate discrete and continuous quantities. Unlike many existing approaches that primarily focus on intracellular mechanisms, multi-level NWN allow for the modeling of both intracellular and supracellular information, enabling a broader scope of representation and effective analysis of complex biological systems [23]. However, the increased expressivity of NWNs comes at the cost of greater model complexity and computational demands for simulation algorithms, a common trade-off in computational modeling [50]. In conclusion, the decision to employ NWN for demonstrating BiSDL stems from the desire to highlight its full expressive capacity in generating complex models. BiSDL may support compilation into various low-level formalisms, generating models on different points across this trade-off.

To support NWN, BiSDL compilation generates models implemented with `nwn-snakes`, a customized version of the `SNAKES` library [51]. `SNAKES` is an efficient Python library for the design and simulation of PN [51]. 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, 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 Availability of data and materials).

The spatial hierarchy underlying BiSDL descriptions (see Sect. 3.2) is translated into a low-level model based on a system of nested spatial grids represented by PN. In this model, the 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 multicellular 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 NWN modeling describing the mapping between BiSDL building blocks and NWN.

BiSDL supports high interpretability of generated NWN models in two ways. Firstly, a compilation of BiSDL constructs labels the resulting low-level constructs with the high-level specific parameters. For instance, the construct `PROTEIN_COMPLEX_FORMATION(3*LuxR_protein, 3*AHL_molecule, 3*LuxR_AHL_complex)` generates NWN constructs containing the product name: `LuxR_AHL_complex`. Secondly, in BiSDL, any construct can be wrapped into a process, and the process is assigned a custom name. This feature supports the direct reuse of processes in general and the reuse of constructs wrapped up in processes by leveraging the process name. Algorithm 4 exemplifies this mechanism: the `PROCESS` defined in lines 18–21 encapsulates `TRANSCRIPTION`, `TRANSLATION`, and `DEGRADATION` constructs, and is named `CD19_production`. The same process is reused (by name) in line 37. Indeed, the `SCOPE` defined in lines 36–40 (5 lines of code) reuses processes defined only once for the first `SCOPE`, which spans over 35 lines of code (1–35). In compilation, to avoid ambiguity, each time the same process is used again in the BiSDL description, a new set of low-level elements is generated and named by appending a progressive number. In the same example from Algorithm 4, the first instance of `PROCESS CD19_production` (lines 18–21) is assigned the name `CD19_production_process_0` in the NWN model. The second instance (line 37) is internally assigned the name `CD19_production_process_1`, and thereafter.

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.

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

Listing 1 `nwn-petrisim` instantiation

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 value ranges.

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.

```
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()
```

Listing 2 `nwn-petrisim` stimuli administration

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 specific transitions from firing.

Results and discussion

The BiSDL allows synthetic biologists to quickly model and design multicellular synthetic systems, simulating their behaviour. 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 [52]. Complex

circuits enable a broader range of controllable behaviours 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 [53], 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 behaviour 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) model description and (2) exploration of system behaviour over three case studies of multicellular synthetic designs: a bacterial consortium (see Sect. 4.1), a synthetic morphogen system (see Sect. 4.2), and a conjugative plasmid transfer (see Sect. 4.3).

Case study 1—bacterial consortium

The first case study focuses on implementing gene expression control across different 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 behaviour. The Registry for Standard Biological Parts holds a collection of predefined genetic constructs with known functionality [54]. These constructs can serve as templates for the DBTL process. Selecting and combining these parts makes it possible to design a bacterial consortium where the overall genetic device enforces LacI-operated gene expression regulation across cells. This consortium comprises two cell types. *Controller cells* establish baseline 3-oxohexanoyl-homoserine lactone (3OC6HSL) production, inhibited by a

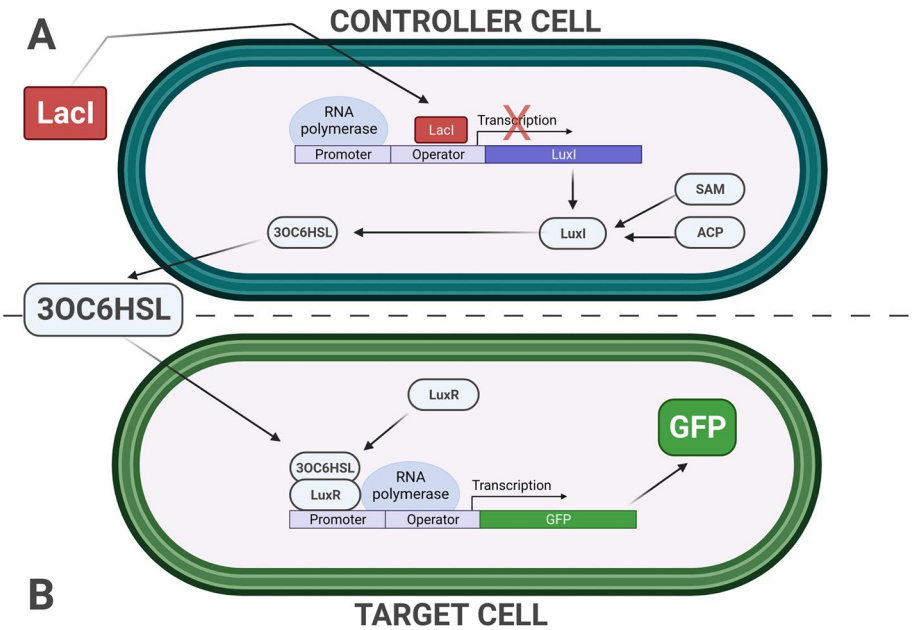

Fig. 4 The multicellular bacterial consortium synthetic design was considered for the first case study. This consortium comprises two cell types. (A) *Controller cells* establish baseline 3OC6HSL production, inhibited by a reference signal (LacI administration); (B) *Target cells* initiate GFP reporter signal production only when receiving the AHL molecular signal, which, in this system, is 3OC6HSL

reference signal: LacI administration (Fig. 4, top panel 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 (Fig. 4, bottom panel 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 [55] constructs 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.

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.

Algorithm 3 BiSDL description of the bacterial consortium.

```
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]
```

The illustrated `bacterialConsortium MODULE` contains two `SCOPE` statements: one for the Controller cell (`producer`) (Algorithm 3, lines 3–20); another one for the Target cell (`sensor`) (Algorithm 3, lines 21–34). Each `SCOPE` definition includes its name and two-dimensional coordinates on the spatial grid underlying the model (Algorithm 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 (Algorithm 3, line 2) indicates the base pace for the `bacterialConsortium`. On the other hand, the `TIMESCALE` of each `PROCESS` (Algorithm 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` 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 (Algorithm 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 files in the public `GitHub` repository, see Availability of data and materials).

To compile the BiSDL description into a `nwn-snakes` model, the synthetic biologist uses the BiSDL compiler (see Sect. 3.4) to generate a `nwn-snakes` file that contains all the NWN models required by the BiSDL description. Visualization of the NWN models relies on the `GraphViz` visualization tool [56], provided by `SNAKES` as a plugin. For this use case, the NWN 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 5 visualizes the top-level NWN model, where the places that contain net tokens correspond to the two BiSDL `SCOPE` statements. The *producer* place holds the

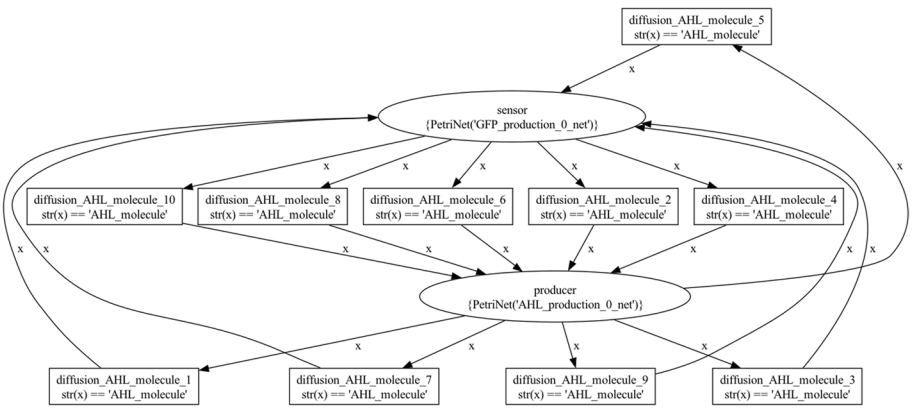

Fig. 5 The top-level bacterial_consortium net architecture. The places that contain net tokens correspond to the two BiSDL `SCOPE` statements. The *producer* place holds the *AHL_production* net token, while the *sensor* place holds the *GFP_production* net token. Several transitions connect the two 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

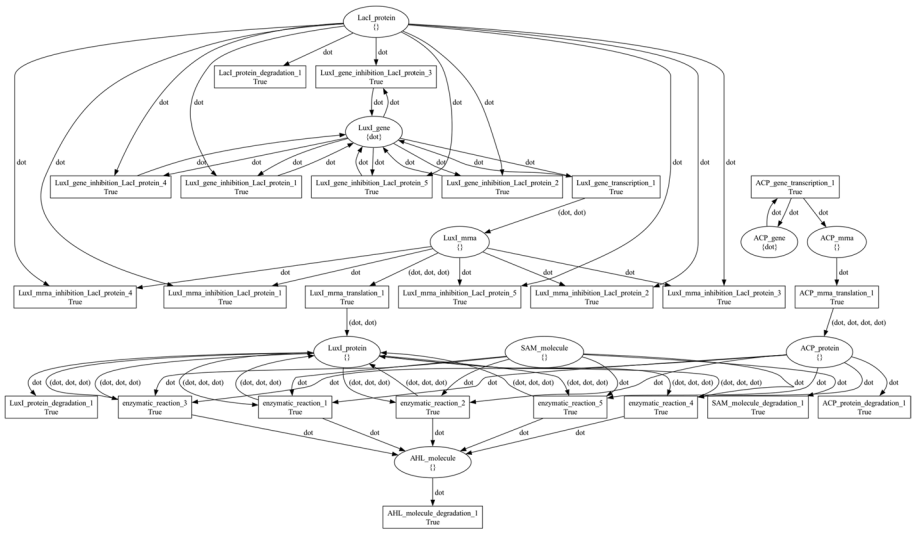

Fig. 6 The producer bacterial_consortium net token architecture. Places model genes, transcripts, proteins, and molecules, while transitions model processes involving them, such as transcription, translation, degradation, and enzymatic reactions. Black tokens model discrete quantities of resources in each place and are represented by the `dot` symbol

AHL_production net token (Fig. 6), while the *sensor* place holds the *GFP_production* net token (Fig. 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 Sect. 3.4.1).

Figure 6 and Figure 7 visualize the *producer* and *sensor* net tokens, respectively. In these PN, 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.

The simulation of BiSDL-compiled `nwn-snakes` models shows that LacI levels control *GFP_protein* levels, consistently with the expected behaviour 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 Sect. 3.4.2).

Figure 8 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 considered LacI administration schemes. *noLacI* (Fig. 8, top left panel A) does not interfere with `AHL_molecule` levels (Fig. 8, middle left panel D), inducing transcription of high `GFP_reporter_protein` readout signals (Fig. 8, bottom left panel G). *lowLacI* (Fig. 8, top central panel B) hampers `AHL_molecule` levels (Fig. 8, middle central panel E), resulting in the transcription of lower

Fig. 7 The sensor bacterial_consortium net token architecture. 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

`GFP_reporter_protein` (Fig. 8, bottom central panel H). *highLacI* (Fig. 8, top right panel C) almost shuts down `AHL_molecule` levels (Fig. 8, middle right panel F), suppressing the transcription of high `GFP_reporter_protein` readout signals almost completely (Fig. 8, bottom right panel I). The results show consistency with the expected system behavior, an inverse relation between LacI stimulus and readout signal levels.

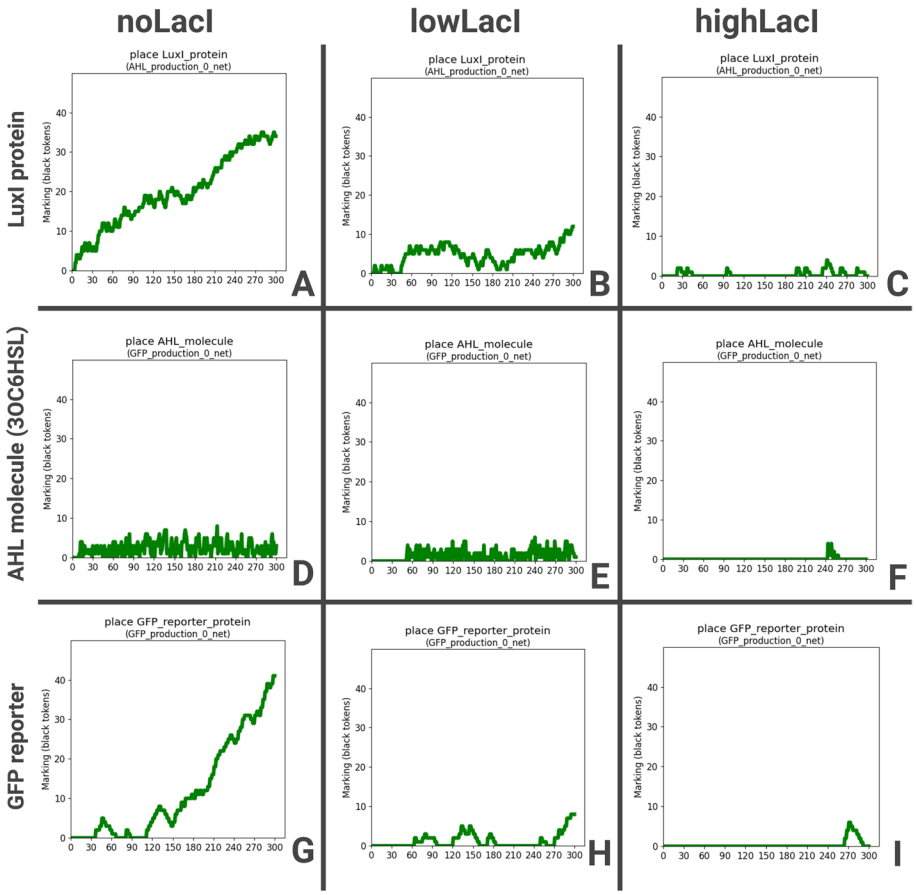

Fig. 8 Marking evolution of the `LacI` signal mediators (`Lux1_protein` and `AHL_molecule`) and Target (`GFP_reporter_protein`) in the bacterial consortium after three LacI administration schemes. (A) *noLacI* does not interfere with (D) `AHL_molecule` levels, (G) inducing transcription of high `GFP_reporter_protein` readout signals. (B) *lowLacI* (E) hampers `AHL_molecule` levels, resulting in (H) the transcription of lower `GFP_reporter_protein`. (C) *highLacI* (F) almost shuts down `AHL_molecule` levels, (I) suppressing the transcription of high `GFP_reporter_protein` readout signals almost completely. The results show consistency with the expected system behavior, an inverse relation between LacI stimulus and readout signal levels

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 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.

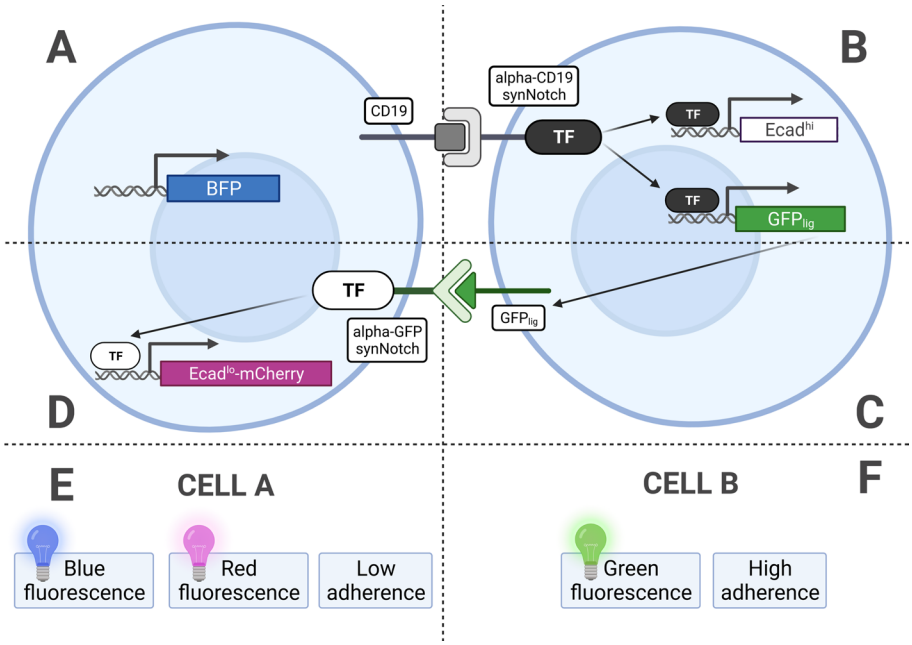

Fig. 9 The multicellular synthetic circuit mediated by cell-cell communication makes the RGB pattern emerge. (**A**) The Sender cells (Cells A) constitutively express blue fluorescent protein (BFP), CD19 ligand, and the anti-GFP synthetic Notch (synNotch) receptor, which drives expression of a low amount of E-cadherin ($Ecad_{lo}$) fused with a (**D**) mCherry reporter for visualization. (**B**) Receiver cells (Cells B) inducibly express E-cadherin ($Ecad_{hi}$) and a modified form of GFP working as a synNotch ligand on the cell membrane (GFP_{lig}). (**C**) GFP_{lig} serves as both a fluorescent reporter and a ligand for a secondary synNotch receptor with the cognate anti-GFP binding domain. (**E**) Cells A have low adherence, blue fluorescence, and inducible red fluorescence; (**F**) Cells B have inducible green fluorescence and high adherence. Adapted from [57]

This time, rather than relying on Standard Parts (refer to Sect. 4.1), the objective is to replicate designs found in the scientific literature, such as the modular synNotch system outlined in [57], 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 intracellular 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. 9, top right panel B) inducibly expresses E-cadherin ($Ecad_{hi}$) and a modified form of GFP working as a synNotch ligand on 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 (Fig. 9, central right panel C). The Sender cells (Cells A, Fig. 9, top left panel A) constitutively express BFP, CD19 ligand, and the anti-GFP synNotch receptor, which drives expression of a low amount of E-cadherin ($Ecad_{lo}$) fused with a mCherry reporter for visualization (Fig. 9, central left panel D). Thus, Cells A have low adherence, blue fluorescence, and inducible red fluorescence (Fig. 9, bottom left panel E), while Cells B have inducible green fluorescence and high adherence (Fig. 9, bottom right panel F).

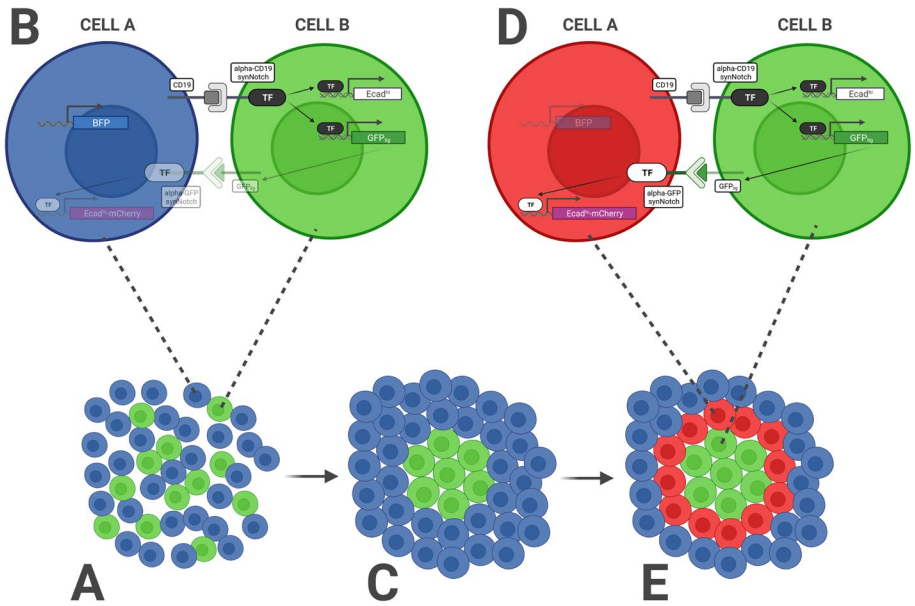

Fig. 10 The RGB synthetic morphogenesis process for the second case study. (**A**) Cells start as a disorganized aggregate; (**B**) CD19 in Cells A activates anti-CD19 synNotch in Cells B, inducing the expression of a high level of E-cadherin ($Ecad_{hi}$) and GFP_{lig}; (**C**) Cells B aggregate and form a compact group in the middle of the aggregate; (**D**) 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, (**E**) making a spatially organized pattern of cells emerge in a synthetic morphogenetic pattern 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).Adapted from [57]

Cells start as a disorganized aggregate (Fig. 10, bottom left panel 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. 10, top left panel B). Cells B thus aggregate and form a compact group in the middle of the aggregate (Fig. 10, bottom central panel 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. 10, top right panel D), and making a spatially organized pattern of cells emerge in a synthetic morphogenetic pattern (Fig. 10, bottom right panel 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 BiSDL description of the RGB synthetic morphogen system.

```
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 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` PN models are not included due to their size, but they are available from the BiSDL `GitHub` public repository (see Availability of data and materials).

BiSDL again supports a compact description of the system composed of 165 lines of code, approximately 9% of the corresponding compiled `nwn-snakes` model file, having 1807 lines of code (based on the files in the public `GitHub` repository, see Availability of data and materials).

Results recapitulate the emergence of the expected simplistic version of one of the synthetic morphogenetic patterns presented in [57]. Fig. 11 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 Algorithm 4). At first (t = 10), the

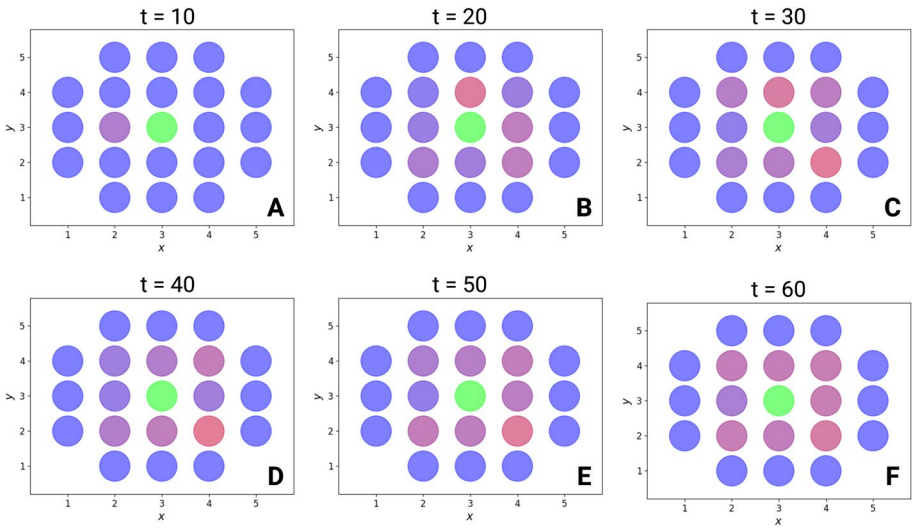

Fig. 11 Evolution of the three fluorescent marker levels (GFP, BFP and mCherry) in each cell on the two-dimensional spatial grid along the simulation of the `nwn-snakes` Python models compiled from the BiSDL description. (A) At first (t = 10), the central cell slightly affects only one of its neighbors of the first degree, while other cells keep producing the BFP signal; (B-E) 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); (F) all first-degree neighbors of the central cell express high levels of mCherry (t=60)

central cell slightly affects only one of its neighbors of the first degree, while other cells keep producing the BFP signal (Fig. 11, top left panel 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, Fig. 11, top central panel B, top right panel C, bottom left panel D, bottom central panel E); at t=60 all first-degree neighbors of the central cell express high levels of mCherry (Fig. 11, bottom right panel 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).

Case study 3—conjugative plasmid transfer

Plasmids are crucial in disseminating antibiotic resistance, virulence genes, and various adaptive traits within bacterial communities through horizontal gene transfer [58]. The third case study examines antibiotic resistance (R) conjugative plasmids transfer between bacterial cells, mirroring the fundamental conjugation mechanism presented in [59]. As depicted in Fig. 12, the plasmid transfer process [60] initiates with a Donor cell harboring a conjugative R plasmid (Fig. 12, top left). The Donor extends a Pilus, a proteinaceous protrusion [61], encoded by the R plasmid, to establish contact with a

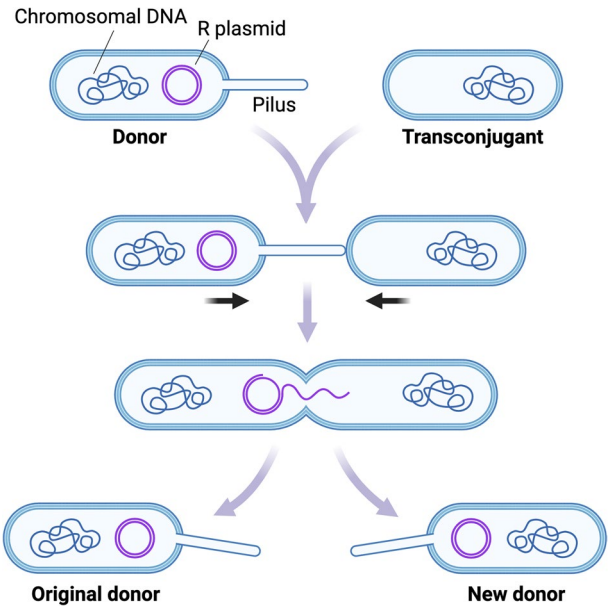

Fig. 12 The mechanism considered for the third case study is antibiotic resistance (R) plasmid transfer across bacterial cells, recapitulating the basic conjugation mechanism modeled in [59]. The plasmid transfer mechanism [60] begins with a Donor cell that carries a conjugative R plasmid (top left). The Donor extends a Pilus, a proteinaceous protrusion [61] to contact a compatible receiver cell, named the Transconjugant cell (top right). In this example, the Pilus is encoded in the R plasmid. Upon contact, the Pilus retracts, pulling the cells together and establishing a conjugation bridge. This bridge enables the transfer of one of the plasmid DNA strands in linearized form from the Donor to the Transconjugant (center). At this point, both cells hold a single-stranded copy of the R plasmid. Finally, both cells build the second strand for their respective R plasmid copies through circularization and DNA synthesis. Thus, both Donor and Transconjugant cells are equipped to disseminate the R plasmid further, making them both Donor cells (bottom) enacting antibiotic resistance propagation

compatible recipient cell, referred to as the Transconjugant cell [59] (Fig. 12, top right). Upon contact, the Pilus retracts, bringing the cells into proximity and forming a conjugation bridge. This bridge facilitates the transfer of one of the plasmid DNA strands in a linearized form from the Donor to the Transconjugant (Fig. 12, center). Subsequently, both cells harbor a single-stranded copy of the R plasmid. Finally, through circularization and DNA synthesis, the Donor and Transconjugant cells complete the second strand for their respective R plasmid copies. Consequently, both cells become capable of further disseminating the R plasmid, effectively functioning as Donor cells (Fig. 12, bottom), thereby facilitating the propagation of antibiotic resistance.

Algorithm 5 provides a BiSDL description of the conjugative R plasmid transfer from the Donor to the Transconjugant cell.

Algorithm 5 BiSDL description of the conjugative R plasmid transfer from the Donor to the Transconjugant cell.

```
1  MODULE plasmidTransfer
2      TIMESCALE 1
3      SCOPE donor (0, 0)
4          PROCESS donor
5              TIMESCALE 2
6              CUSTOM_PROCESS ([circular_plasmid_molecule], [R_mrna, circular_plasmid_molecule])
7              TRANSLATION (R_mrna, R_protein)
8              CUSTOM_PROCESS ([circular_plasmid_molecule], [pilus_mrna, circular_plasmid_molecule])
9              TRANSLATION (pilus_mrna, pilus_protein)
10             CUSTOM_PROCESS ([circular_plasmid_molecule, 2*pilus_protein],
                 [circular_plasmid_molecule, linear_plasmid_molecule])
11             DEGRADATION (R_mrna)
12             DEGRADATION (pilus_mrna)
13             DEGRADATION (R_protein)
14             DEGRADATION (pilus_protein)
15         JUXTACRINE_SIGNAL linear_plasmid_molecule -> pilus
16     SCOPE pilus (0, 1)
17         PROCESS pilus
18             TIMESCALE 4
19             CUSTOM_PROCESS ([linear_plasmid_molecule], [linear_plasmid_molecule])
20         JUXTACRINE_SIGNAL linear_plasmid_molecule -> transconjugant
21     SCOPE transconjugant (0, 2)
22         PROCESS transconjugant
23             TIMESCALE 2
24             CUSTOM_PROCESS ([linear_plasmid_molecule], [circular_plasmid_molecule])
25             CUSTOM_PROCESS ([circular_plasmid_molecule], [R_mrna, circular_plasmid_molecule])
26             TRANSLATION (R_mrna, R_protein)
27             CUSTOM_PROCESS ([circular_plasmid_molecule], [pilus_mrna, circular_plasmid_molecule])
28             TRANSLATION (pilus_mrna, pilus_protein)
29             DEGRADATION (R_mrna)
30             DEGRADATION (pilus_mrna)
31             DEGRADATION (R_protein)
32             DEGRADATION (pilus_protein)
```

The illustrated `plasmidTransfer` `MODULE` contains three `SCOPE` statements: one for the Donor cell (`donor`, (Algorithm 5, lines 3–15); another one for the Transconjugant cell (`transconjugant`, Algorithm 5, lines 21–32), and a third one for the Pilus proteinaceous structure mediating the conjugation process (`pilus`, Algorithm 5, lines 16–20). This shows the flexibility of the `SCOPE` construct in supporting the modeler for expressing different types of biological compartments. `JUXTACRINE_SIGNAL` mechanisms connect the `SCOPE` statements from the `donor` to the `transconjugant` through the `pilus`, recapitulating the R plasmid transfer process.

Figure 13 visualizes the top-level NWN model for this use case, having a place for each of the three BiSDL `SCOPE` statements, holding the *donor* net token (Fig. 14), the *transconjugant* net token (Fig. 15) and the *pilus* net token (Fig. 16), respectively.

The simulation of BiSDL-compiled `nwn-snakes` models demonstrates that transferring R plasmids from the Donor to the Transconjugant cells aligns with anticipated behavior. Throughout the simulation, the Donor cell consistently retains one R plasmid (Fig. 17, top left), while the Transconjugant cell initially lacks any R plasmids (Fig. 17, top right). The R plasmid within the Donor encodes the Pilus protein, which initiates the conjugation process upon translation. The Pilus mediates two R plasmid transfer events, illustrated by linearized single-strand R plasmids within it (Fig. 17, center), resulting in two subsequent increases in R plasmid copies within the Donor cell (Fig. 17, top right). R plasmids encode the R protein, conferring antibiotic resistance. The R protein is consistently present in the Donor cell due to the R plasmid (Fig. 17, bottom left). Conversely, in the Transconjugant cell, R protein levels only rise above zero after the initial R plasmid transfer event (Fig. 17, bottom right). These findings indicate that the simulation accurately reproduces the conjugative plasmid transfer process, leading to R protein-mediated antibiotic resistance propagation.

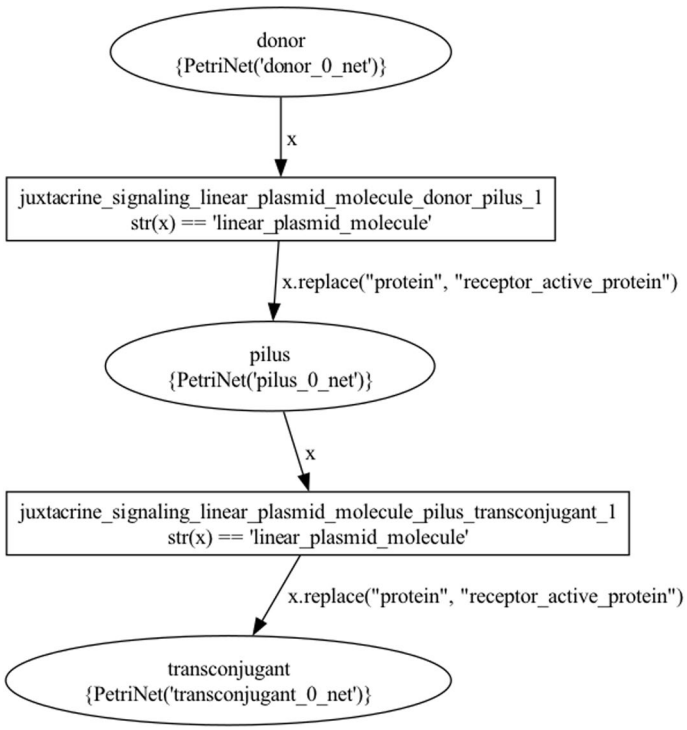

Fig. 13 The top-level plasmid_transfer net architecture. The places that contain net tokens correspond to the two BiSDL `SCOPE` statements, holding the *donor* net token (Fig. 14), the *transconjugant* net token (Fig. 15) and the *pilus* net token (Fig. 16), respectively

Conclusions

In conclusion, the BiSDL framework represents a significant advancement in 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 more accessible than a low-level language. In fact, BiSDL requires only basic modeling and programming skills compared to the advanced modeling and programming skills required by low-level languages. The hierarchical and modular structure of BiSDL is ideal for capturing the inherent complexity of biological systems and constructing reusable and adaptable components.

Through the development of a prototype BiSDL compiler, models can be compiled into low-level descriptions that encapsulate the spatial, hierarchical, and dynamic behaviors of biological entities, using the NWN approach to handle biological complexity. As BiSDL closely mirrors the domain-specific language used within the biological domain, the compiler closes the gap between high-level biological semantics and NWN low-level formalism syntax. Generated models rely on the `nwn-snakes` library, an extension of the `SNAKES` library [51] to support the NWN formalism. Results indicate that BiSDL can dramatically simplify complex model descriptions, significantly reducing the code needed to represent sophisticated systems. This is evidenced by the bacterial consortium, RGB and conjugative plasmid transfer case studies.

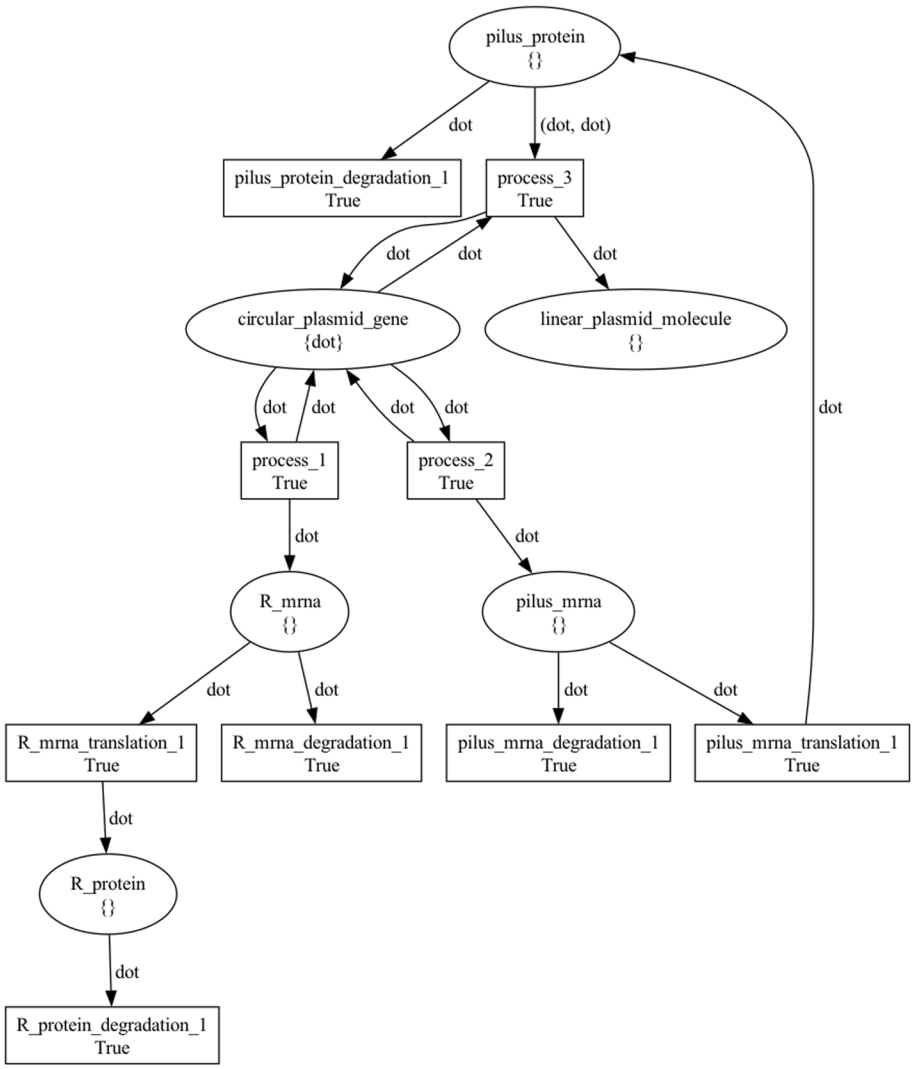

Fig. 14 The *donor* plasmid_transfer net token architecture. Places model genes, transcripts, proteins, and molecules, while transitions model processes involving them, such as transcription, translation, degradation, and enzymatic reactions. Black tokens model discrete quantities of resources in each place and are represented by the `dot` symbol

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

Future developments may integrate into BiSDL a tool for formal verification [62], inherently supported by PN as a low-level formalism. This would allow the validation of complex models by rigorously checking for correctness according to specific criteria. Several integrable tools exist for PN formal verification, including TINA (Time Petri Net Analyzer) [63], ABCD plus Neco for SNAKES models [64] and GreatSPN [65], which provides a user-friendly visual interface, in alignment with BiSDL's aim for broader accessibility.

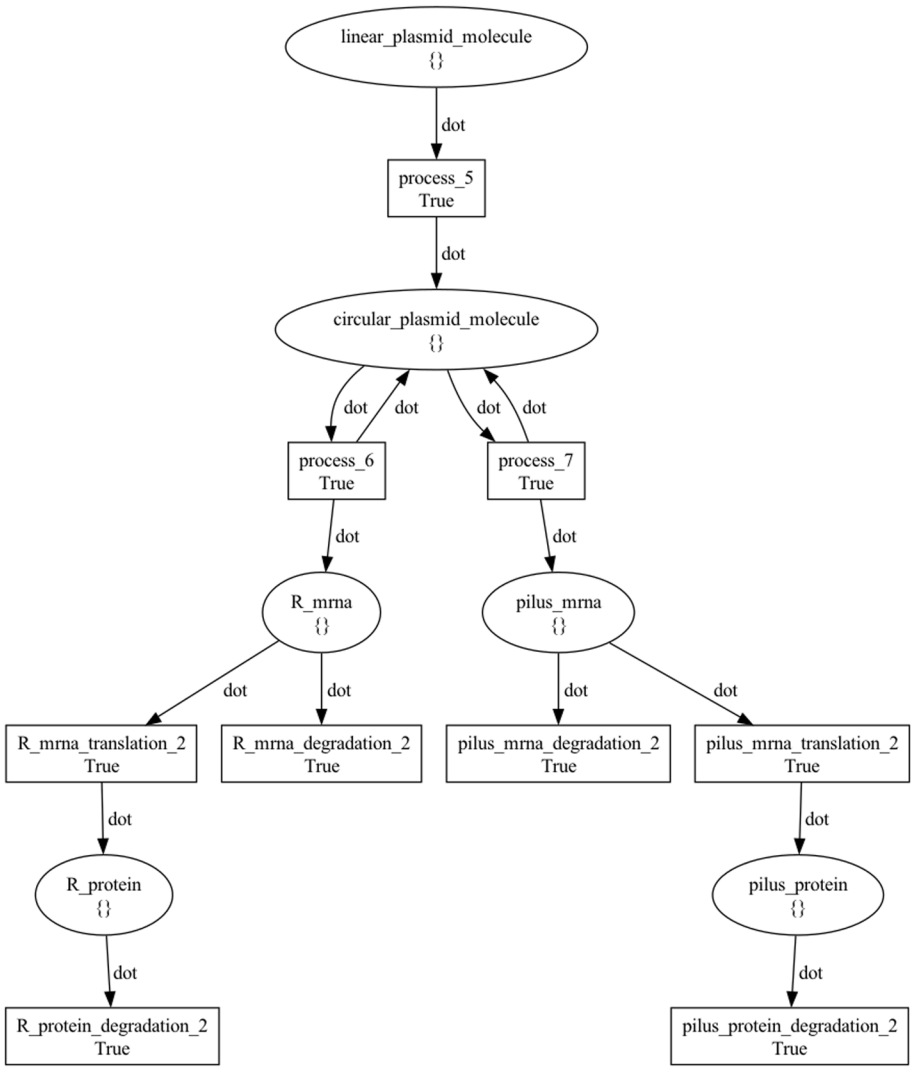

Fig. 15 The *transconjugant* plasmid_transfer net token architecture. Places model genes, transcripts, proteins, and molecules, while transitions model processes involving them, such as transcription, translation, degradation, and enzymatic reactions. Black tokens model discrete quantities of resources in each place and are represented by the `dot` symbol

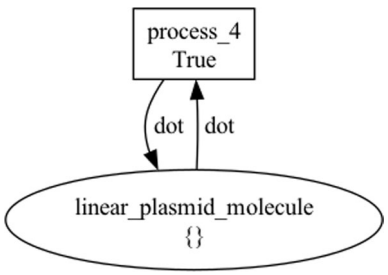

Fig. 16 The *Pilus* plasmid_transfer net token architecture. Places model genes, transcripts, proteins, and molecules, while transitions model processes involving them, such as transcription, translation, degradation, and enzymatic reactions. The black tokens model discrete quantities of resources in each place and are represented by the `dot` symbol

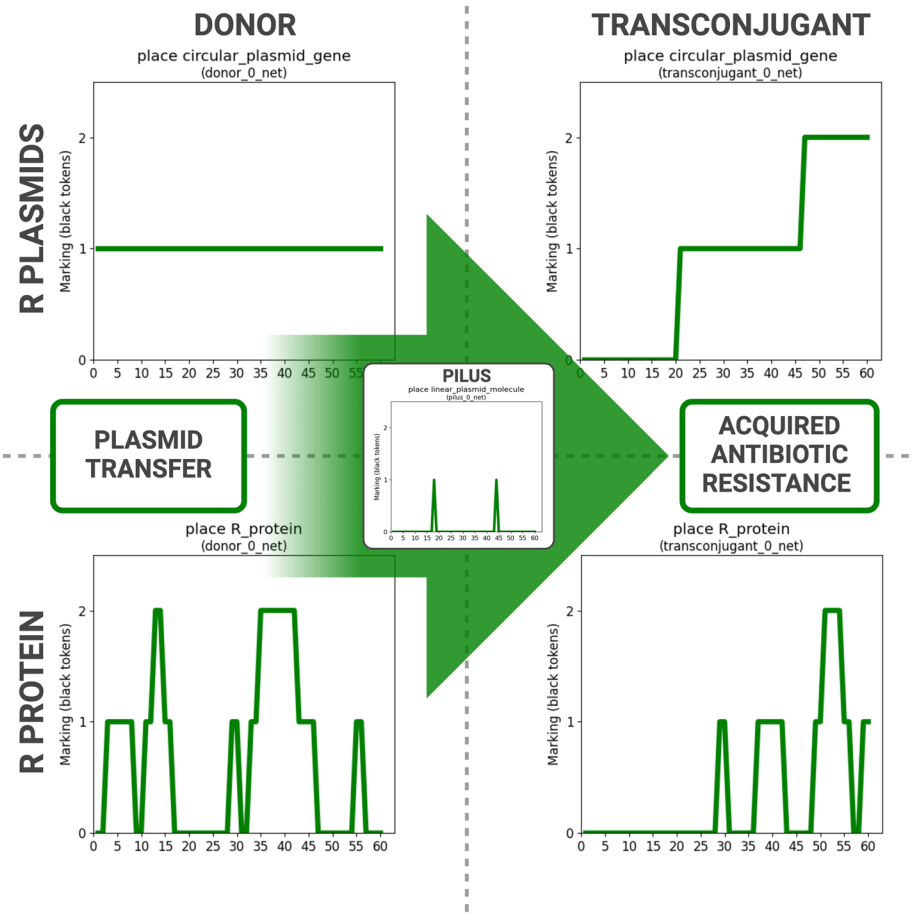

Fig. 17 Simulation of BiSDL-compiled `nwn-snakes` models shows that the transfer of R plasmids from the Donor to the Transconjugant cells is consistent with the expected behavior. The Donor cell holds one R plasmid throughout the simulation (top left), while the Transconjugant cell starts with none (top right). The R plasmid in the Donor encodes for the Pilus protein, which initiates the conjugation process as soon as it is translated. The Pilus mediates two R plasmid transfer events, depicted as the presence of a linearized single-strand R plasmid within it (center), resulting in two subsequent increases of R plasmid copies in the Donor cell (top right). R plasmids encode for R protein, which provides antibiotic resistance. In the Donor cell, the R protein is present from the start due to the presence of the R plasmid (bottom left). Otherwise, protein levels rise above zero in the Transconjugant cell R only after the first R plasmid transfer event (bottom right). These results show that simulation recapitulates the plasmid-transfer conjugation process, causing the acquisition of R protein-mediated antibiotic resistance

The proposed BiSDL constructs exemplify the expressiveness of the language, and its closeness to the biological semantics. Future developments include gradually extending the syntax of the language with additional constructs.

In the future, plans are in place to further enhance BiSDL usability by developing a visual language interface and refining the user interface to simplify combining or editing modules. While, in its current implementation, the BiSDL still requires basic programming and modeling skills, a graphical interface would pave the way to the extensive broadening of the user base to people with no programming skills.

Considering the importance of BiSDL adhering to Findability, Accessibility, Interoperability, and Reuse (FAIR) guidelines for data management and stewardship, adoption and enforcement of a standardized naming convention in BiSDL is of paramount

goal in the future. In this regard, BiSDL current framework is poised for advancements in two critical areas. Firstly, the versatility of its source-to-source compiler, which currently facilitates the translation of BiSDL into NWN models, could be expanded to support additional formalisms and standards, including those under the COMBINE initiative. Secondly, future works may integrate an ONTOLOGY_ID field into the language within the Metadata representation SBO category to encourage adherence to standards and foster model exchange.

Abbreviations

NWN	Nets-Within-Nets
PN	Petri Nets
BiSDL	Biology System Description Language
DBTL	Design-Build-Test-Learn
VHDL	Very High-Speed Integrated Circuit Hardware Description Language
AHL	Acyl-homoserine lactones
3OC6HSL	3-Oxohexanoyl-homoserine lactone
GFP	Green Fluorescent Protein
ACP	Acylated acyl carrier protein
SAM	S-adenosylmethionine
LacI	Lactose repressor protein
synNotch	Synthetic Notch
BFP	Blue fluorescent protein
DSL	Domain-Specific Language
GEC	Genetic Engineering of Cells
GUBS	Genomic Unified Behavior Specification
SBOL	Synthetic Biology Open Language
COMBINE	COmputational Modelling in Blology NEtwork
SBGN	Systems Biology Graphical Notation
SBML	Systems Biology Markup Language
ODE	Ordinary Differential Equations
OOP	Object-Oriented Programming
RGB	Red, green, and blue
IBL	Infobiotics Language
SED-ML	Simulation Experiment Description Markup Language
gro	The Cell Programming Language
SBO	Systems Biology Ontology
FAIR	Findability, Accessibility, Interoperability, and Reuse
GUI	Graphical User Interface

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-024-05782-x.

Additional file 1: BiSDL Modules Library. Appendix illustrating the syntax and semantics of the basic building blocks used to create a Biology System Description Language (BiSDL) description (Section 1) and the mapping between BiSDL constructs and the chosen NWN representation (Section 2).

Acknowledgements
Not applicable.

Author 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.

Funding
This study was carried out within the project "SAISEI—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].

Availability of data and materials
The code and data supporting this study and required to reproduce all published results are publicly available on GitHub at https://github.com/smilies-polito/BiSDL/tree/v1.2 and referenced with Zenodo at https://zenodo.org/doi/10.5281/zenodo.10895847.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no conflict of interest.

Received: 11 January 2024 Accepted: 12 April 2024
Published online: 25 April 2024

References

1. St. John PC, Bomble YJ. Approaches to computational strain design in the multiomics era. Front Microbiol. 2019;10:597. https://doi.org/10.3389/fmicb.2019.00597.
2. Chen Y, Banerjee D, Mukhopadhyay A, Petzold CJ. Systems and synthetic biology tools for advanced bioproduction hosts. Curr Opin Biotechnol. 2020;64:101–9. https://doi.org/10.1016/j.copbio.2019.12.007
3. McCarty NS, Ledesma-Amaro R. Synthetic biology tools to engineer microbial communities for biotechnology. Trends Biotechnol. 2018. https://doi.org/10.1016/j.tibtech.2018.11.002
4. Wu G, Yan Q, Jones JA, Tang YJ, Fong SS, Koffas MA. Metabolic burden: cornerstones in synthetic biology and metabolic engineering applications. Trends Biotechnol. 2016;34(8):652–64. https://doi.org/10.1016/j.tibtech.2016.02.010
5. Borkowski O, Ceroni F, Stan G-B, Ellis T. Overloaded and stressed: whole-cell considerations for bacterial synthetic biology. Curr Opin Microbiol. 2016;33:123–30. https://doi.org/10.1016/j.mib.2016.07.009
6. Ebrahimkhani MR, Levin M. Synthetic living machines: a new window on life. Iscience. 2021;24(5): 102505. https://doi.org/10.1016/j.isci.2021.102505
7. Martini L, Amprimo G, Di Carlo S, Olmo G, Ferraris C, Savino A, Bardini R. Neuronal Spike Shapes (NSS): A straightforward approach to investigate heterogeneity in neuronal excitability states. Comput Biol Med. 2024;168:107783. https://doi.org/10.1016/j.compbiomed.2023.107783
8. Martini L, Bardini R, Savino A, Di Carlo S. GAGAM v1. 2: An improvement on peak labeling and genomic annotated gene activity matrix construction. Genes. 2022;14(1):115. https://doi.org/10.3390/genes14010115
9. Bardini R, Politano G, Benso A, Carlo SD. Using multi-level petri nets models to simulate microbiota resistance to antibiotics. In: 2017 IEEE international conference on bioinformatics and biomedicine (BIBM), pp. 128–133; 2017. https://doi.org/10.1109/BIBM.2017.8217637.
10. Bardini R, Di Carlo S, Politano G, Benso A. Modeling antibiotic resistance in the microbiota using multi-level Petri Nets. BMC Syst Biol. 2018;12:59–79. https://doi.org/10.1186/s12918-018-0627-1.
11. Bardini R, Di Carlo S. Computational methods for biofabrication in tissue engineering and regenerative medicine-a literature review. Comput Struct Biotechnol J. 2024. https://doi.org/10.1016/j.csbj.2023.12.035.
12. Giannantoni L, Bardini R, Di Carlo S. A Methodology for co-simulation-based optimization of biofabrication protocols. In: Rojas I, Valenzuela O, Rojas F, Herrera LJ, Ortuño F, editors. Bioinformatics and biomedical engineering. IWBBIO 2022. Lecture Notes in Computer Science, vol 13347. Cham: Springer; 2022. https://doi.org/10.1007/978-3-031-07802-6_16.
13. Giannantoni L, Savino A, Di Carlo S. Optimization of synthetic oscillatory biological networks through reinforcement learning. In: 2023 IEEE international conference on bioinformatics and biomedicine (BIBM), Istanbul, 2023. IEEE. https://doi.org/10.1109/bibm58861.2023.10385777.
14. Bardini R, Politano G, Benso A, Di Carlo S. Multi-level and hybrid modelling approaches for systems biology. Comput Struct Biotechnol J. 2017. https://doi.org/10.1016/j.csbj.2017.07.005.
15. Bardini R, Politano G, Benso A, Di Carlo S. Computational tools for applying multi-level models to synthetic biology. In: Singh S, editors. Synthetic Biology. Singapore: Springer. https://doi.org/10.1007/978-981-10-8693-9_7
16. Pouvreau B, Vanhercke T, Singh S. From plant metabolic engineering to plant synthetic biology: the evolution of the design/build/test/learn cycle. Plant Sci. 2018;273:3–12. https://doi.org/10.1016/j.plantsci.2018.03.035.
17. Murphy F. Open access, open data, FAIR Data and their implications for life sciences researchers. Emerging Top Life Sci. 2018;2(6):759–62. https://doi.org/10.1042/etls20180163
18. Stall S, Yarmey L, Cutcher-Gershenfeld J, Hanson B, Lehnert K, Nosek B, Parsons M, Robinson E, Wyborn L. Make scientific data FAIR. Nat Publ Group. 2019. https://doi.org/10.1038/d41586-019-01720-7.
19. Keating SM, Waltemath D, König M, Zhang F, Dräger A, Chaouiya C, Bergmann FT, Finney A, Gillespie CS, Helikar T, et al. SBML Level 3: an extensible format for the exchange and reuse of biological models. Mol Syst Biol. 2020;16(8):9110. https://doi.org/10.15252/msb.20199110.

20. Gleeson P, Crook S, Cannon RC, Hines ML, Billings GO, Farinella M, Morse TM, Davison AP, Ray S, Bhalla US, et al. NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail. PLoS Comput Biol. 2010;6(6):1000815. https://doi.org/10.1371/journal.pcbi.1000815.
21. Smith LP, Bergmann FT, Garny A, Helikar T, Karr J, Nickerson D, Sauro H, Waltemath D, König M. The simulation experiment description markup language (SED-ML): language specification for level 1 version 4. J Integr Bioinform. 2021;18(3):20210021. https://doi.org/10.1515/jib-2021-0021.
22. Konur S, Mierla L, Fellermann H, Ladroue C, Brown B, Wipat A, Twycross J, Dun BP, Kalvala S, Gheorghe M, et al. Toward full-stack in silico synthetic biology: integrating model specification, simulation, verification, and biological compilation. ACS Synth Biol. 2021;10(8):1931–45. https://doi.org/10.1021/acssynbio.1c00143.
23. Bartocci E, Lió P. Computational modeling, formal analysis, and tools for systems biology. PLoS Comput Biol. 2016;12(1):1004591. https://doi.org/10.1371/journal.pcbi.1004591.
24. Bardini R, Benso A, Di Carlo S, Politano G, Savino A (2016) Using nets-within-nets for modeling differentiating cells in the epigenetic landscape. In: Ortuño F, Rojas I, editors. Bioinformatics and biomedical engineering. IWBBIO 2016. Lecture Notes in Computer Science, vol 9656. Cham: Springer. https://doi.org/10.1007/978-3-319-31744-1_28.
25. Bardini R, Benso A, Politano G, Di Carlo S. Nets-within-nets for modeling emergent patterns in ontogenetic processes. Comput Struct Biotechnol J. 2021;19:5701–21. https://doi.org/10.1016/j.csbj.2021.10.008.
26. Muggianu F, Benso A, Bardini R, Hu E, Politano G, Di Carlo S. Modeling biological complexity using biology system description language (BiSDL). In: 2018 IEEE international conference on bioinformatics and biomedicine (BIBM), pp. 713–717, 2018. IEEE. https://doi.org/10.1109/bibm.2018.8621533.
27. Valk R. Object petri nets: using the nets-within-nets paradigm. Berlin: Springer; 2004. p. 819–48. https://doi.org/10.1007/978-3-540-27755-2_23.
28. Bardini R, et al. A diversity-aware computational framework for systems biology. Ph.D. thesis, Politecnico di Torino, 2019 (cit. on p. 5), 2019. https://hdl.handle.net/11583/2752792.
29. Galdzicki M, Clancy KP, Oberortner E, Pocock M, Quinn JY, Rodriguez CA, Roehner N, Wilson ML, Adam L, Anderson JC, et al. The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. Nat Biotechnol. 2014;32(6):545. https://doi.org/10.1038/nbt.2891.
30. Cox RS, Madsen C, McLaughlin JA, Nguyen T, Roehner N, Bartley B, Beal J, Bissell M, Choi K, Clancy K, et al. Synthetic biology open language (SBOL) version 2.2.0. J Integr Bioinform 2018;15(1). https://doi.org/10.1515/jib-2018-0001.
31. Chaouiya C, Bérenguier D, Keating SM, Naldi A, Van Iersel MP, Rodriguez N, Dräger A, Büchel F, Cokelaer T, Kowal B, et al. SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools. BMC Syst Biol. 2013;7(1):135. https://doi.org/10.1186/1752-0509-7-135.
32. Bartley BA, Choi K, Samineni M, Zundel Z, Nguyen T, Myers CJ, Sauro HM. pySBOL: a python package for genetic design automation and standardization. ACS Synth Biol. 2018;8(7):1515–8. https://doi.org/10.1021/acssynbio.8b00336.s001.
33. Bornstein BJ, Keating SM, Jouraku A, Hucka M. LibSBML: An API library for SBML. Bioinformatics. 2008;24(6):880–1. https://doi.org/10.1093/bioinformatics/btn051.
34. Hucka M, Nickerson DP, Bader GD, Bergmann FT, Cooper J, Demir E, Garny A, Golebiewski M, Myers CJ, Schreiber F, Waltemath D, Le NovÃšre N. Promoting coordinated development of community-based information standards for modeling in biology: the COMBINE initiative. Front Bioeng Biotechnol. 2015;3:19. https://doi.org/10.3389/fbioe.2015.00019.
35. COMBINE: The COMBINE standards. [Online] https://co.mbine.org/standards 2018.
36. Waltemath D, Golebiewski M, Blinov ML, Gleeson P, Hermjakob H, Hucka M, Inau ET, Keating SM, König M, Krebs O, et al. The first 10 years of the international coordination network for standards in systems and synthetic biology (COMBINE). J Integr Bioinform. 2020;17(2–3):20200005. https://doi.org/10.1515/jib-2020-0005.
37. Demir E, Cary MP, Paley S, Fukuda K, Lemer C, Vastrik I, Wu G, D'eustachio P, Schaefer C, Luciano J, et al. The biopax community standard for pathway data sharing. Nat Biotechnol. 2010;28(9):935–42. https://doi.org/10.1038/nbt.1666.
38. Le Novere N, Hucka M, Mi H, Moodie S, Schreiber F, Sorokin A, Demir E, Wegner K, Aladjem MI, Wimalaratne SM, et al. The systems biology graphical notation. Nat Biotechnol. 2009;27(8):735. https://doi.org/10.1038/nbt.1558.
39. Lloyd CM, Halstead MDB, Nielsen PF. CellML: its future, present and past. Prog Biophys Mol Biol. 2004;85(2):433–50. https://doi.org/10.1016/j.pbiomolbio.2004.01.004.
40. Köhn D, Le Novère N. SED-ML—an XML format for the implementation of the MIASE guidelines. In: Heiner M, Uhrmacher AM, editors. Computational methods in systems biology. Berlin: Springer; 2008. p. 176–90. https://doi.org/10.1007/978-3-540-88562-7_15.
41. Schölzel C, Blesius V, Ernst G, Dominik A. Characteristics of mathematical modeling languages that facilitate model reuse in systems biology: a software engineering perspective. NPJ Syst Biol Appl. 2021;7(1):1–20. https://doi.org/10.1038/s41540-021-00182-w.
42. Choi K, Medley JK, König M, Stocking K, Smith L, Gu S, Sauro HM. Tellurium: an extensible python-based modeling environment for systems and synthetic biology. Biosystems. 2018;171:74–9. https://doi.org/10.1016/j.biosystems.2018.07.006.
43. Gutierrez M, Gregorio-Godoy P, Pulgar G, Munoz LE, Sáez S, Rodríguez-Patón A. A New Improved and Extended Version of the Multicell Bacterial Simulator gro. ACS Synth Biol. 2017;6(8):1496–508. https://doi.org/10.1021/acssynbio.7b00003.

44. Bilitchenko L, Liu A, Cheung S, Weeding E, Xia B, Leguia M, Anderson JC, Densmore D. Eugene-A domain specific language for specifying and constraining synthetic biological parts, devices, and systems. PLoS ONE. 2011;6(4):18882. https://doi.org/10.1371/journal.pone.0018882.
45. Pedersen M, Phillips A. Towards programming languages for genetic engineering of living cells. J R Soc Interface. 2009;6(suppl-4):437–50. https://doi.org/10.1098/rsif.2008.0516.focus.
46. Basso-Blandin A, Delaplace F. GUBS, a behaviour-based language for design in synthetic biology. Sci Ann Comput Sci. 2013. https://doi.org/10.7561/sacs.2013.1.1.
47. Galdzicki M, Rodriguez C, Chandran D, Sauro HM, Gennari JH. Standard biological parts knowledgebase. PLoS ONE. 2011;6(2):17005. https://doi.org/10.1371/journal.pone.0017005.
48. Kienhuis B, Deprettere EF, Wolf P, Vissers K. A Methodology to design programmable embedded systems. In: International workshop on embedded computer systems, pp. 18–37, 2001. Springer. https://doi.org/10.1007/3-540-45874-3_2.
49. Courtot M, Juty N, Knüpfer C, Waltemath D, Zhukova A, Dräger A, Dumontier M, Finney A, Golebiewski M, Hastings J, et al. Controlled vocabularies and semantics in systems biology. Mol Syst Biol. 2011;7(1):543. https://doi.org/10.1038/msb.2011.77.
50. Medvedev P. Modeling biological problems in computer science: a case study in genome assembly. Brief Bioinform. 2019;20(4):1376–83. https://doi.org/10.1093/bib/bby003.
51. Pommereau F. SNAKES: a flexible high-level petri nets library (tool paper). In: International conference on applications and theory of petri nets and concurrency, pp. 254–265, 2015. Springer. https://doi.org/10.1007/978-3-319-19488-2_13.
52. Purnick PE, Weiss R. The second wave of synthetic biology: from modules to systems. Nat Rev Mol Cell Biol. 2009;10(6):410–22. https://doi.org/10.1038/nrm2698.
53. Amos M, Goñi-Moreno A. Cellular computing and synthetic biology. In: Computational matter, pp. 93–110. Springer, Berlin; 2018. https://doi.org/10.1007/978-3-319-65826-1_7.
54. iGEM: iGEM Parts Registry—Registry of Standard Biological Parts. http://parts.igem.org/ Accessed 07 August 2023.
55. Schaefer AL, Val DL, Hanzelka BL, Cronan JE, Greenberg EP. Generation of cell-to-cell signals in quorum sensing: acyl homoserine lactone synthase activity of a purified Vibrio fischeri LuxI protein. Proc Natl Acad Sci. 1996;93(18):9505–9. https://doi.org/10.1073/pnas.93.18.9505.
56. Ellson J, Gansner E, Koutsofios L, North SC, Woodhull G. Graphviz–open source graph drawing tools. In: Graph drawing: 9th international symposium, GD 2001 Vienna, Austria, September 23–26, 2001 Revised Papers 9, pp. 483–484, 2002. Springer. https://doi.org/10.1007/3-540-45848-4_57.
57. Toda S, Blauch LR, Tang SKY, Morsut L, Lim WA. Programming self-organizing multicellular structures with synthetic cell-cell signaling. Science. 2018;361(6398):156–62. https://doi.org/10.1126/science.aat0271
58. Redondo-Salvo S, Fernández-López R, Ruiz R, Vielva L, Toro M, Rocha EP, Garcillán-Barcia MP, Cruz F. Pathways for horizontal gene transfer in bacteria revealed by a global map of their plasmids. Nat Commun. 2020;11(1):3602. https://doi.org/10.1038/s41467-020-17278-2.
59. Gregory R, Saunders J, Saunders V. Rule-based modelling of conjugative plasmid transfer and incompatibility. Biosystems. 2008;91(1):201–15. https://doi.org/10.1016/j.biosystems.2007.09.003.
60. Virolle C, Goldlust K, Djermoun S, Bigot S, Lesterlin C. Plasmid transfer by conjugation in gram-negative bacteria: from the cellular to the community level. Genes. 2020;11(11):1239. https://doi.org/10.3390/genes11111239.
61. Shanmugasundarasamy T, Govindarajan DK, Kandaswamy K. A review on pilus assembly mechanisms in gram-positive and gram-negative bacteria. Cell Surface. 2022;8: 100077. https://doi.org/10.1016/j.tcsw.2022.100077.
62. Chiola G, Dutheillet C, Franceschinis G, Haddad S. Stochastic well-formed colored nets and symmetric modeling applications. IEEE Trans Comput. 1993;42(11):1343–60. https://doi.org/10.1109/12.247838.
63. Berthomieu B, Diaz M. Modeling and verification of time dependent systems using time petri nets. IEEE Trans Software Eng. 1991;17(3):259. https://doi.org/10.1109/32.75415.
64. Fronc L, Pommereau F. Building Petri nets tools around Neco compiler. In: International workshop on petri nets and software engineering (PNSE 2013), p. 2013. https://hal.science/hal-00911714.
65. Amparore EG, Balbo G, Beccuti M, Donatelli S, Franceschinis G. 30 years of greatspn. Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi on his 70th Birthday, 227–254, 2016. https://doi.org/10.1007/978-3-319-30599-8_9.

Publisher's Note

Leonardo Giannantoni is a Ph.D. candidate at the Department of Control and Computer Engineering of Politecnico di Torino (Italy). 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 (Italy). She holds a Ph.D. (2019) in Control and Computer Engineering from the Politecnico di Torino and an M.Sc. in Molecular Biotechnology (2014) from Università di Torino. Dr. Bardini's 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 is a Full Professor in the Department of Control and Computer Engineering 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. Prof. Di Carlo's research contributions include biological network analysis and simulation, machine learning, image processing, evolutionary algorithms, and development of resilient computer architectures.