[33] Robin Milner. 1980. A Calculus of Communicating Systems. In *Lecture Notes in Computer Science*, Vol. 92.
[34] Nelson Minar, Roger Burkhart, Chris Langton, Manor Askenazi, et al. 1996. The swarm simulation system: A toolkit for building multi-agent simulations. (1996).
[35] M. Nadini, L. Zino, A. Rizzo, and M. Porfiri. 2020. A multi-agent model to study epidemic spreading and vaccination strategies in an urban-like environment. *Applied Network Science* 5, 1 (2020). www.scopus.com Cited By :7.
[36] Marzio Alfio Pennisi. 2012. A mathematical model of immune-system-melanoma competition. *Comput Math Methods Med.* (2012), 850754:1–850754:13. https://doi.org/10.1155/2012/850754
[37] L. Perez and et al. 2009. An agent-based approach for modeling dynamics of contagious disease spread. *International Journal of Health Geographics* 8, 1 (2009).
[38] S. Pernice, P. Castagno, L. Marcotulli, M.M. Maule, L. Richiardi, G. Moirano, M. Sereno, F. Cordero, and M. Beccuti. 2020. Impacts of reopening strategies for COVID-19 epidemic: a modeling study in Piedmont region. *BMC Infectious Diseases* 20, 1 (2020). https://doi.org/10.1186/s12879-020-05490-w
[39] S. Pernice and et al. 2019. A computational approach based on the Colored Petri Net formalism for studying Multiple Sclerosis. *BMC bioinformatics* 20, 6 (2019), 1–17.
[40] S. Pernice, L. Follia, A. Maglione, M. Pennisi, F. Pappalardo, F. Novelli, M. Clerico, M. Beccuti, F. Cordero, and S. Rolla. 2020. Computational modeling of the immune response in Multiple Sclerosis using Epimod framework. *BMC bioinformatics* 21, 17 (2020), 1–20.
[41] Alessia Pini and Simone Vantini. 2013. The interval testing procedure: inference for functional data controlling the family wise error rate on intervals. *MOX-Report* 13 (2013), 2013.
[42] Oliver Reinhardt, Tom Warnke, and Adelinde M Uhrmacher. 2022. A language for agent-based discrete-event modeling and simulation of linked lives. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 32, 1 (2022), 1–26.
[43] Eberhard O Voit, Harald A Martens, and Stig W Omholt. 2015. 150 years of the mass action law. *PLoS computational biology* 11, 1 (2015), e1004012.
[44] U. . Wilensky. 1999. NetLogo. http://ccl.northwestern.edu/netlogo/

A ONLINE RESOURCES

The ESSN models and files to reproduce the analysis showed in this work are freely available at https://github.com/qBioTurin/AB-ESSN. The GreatMod framework is available at https://qbioturin.github.io/epimod/. The last version of GreatSPN framework is available at https://github.com/greatspn/SOURCES. An Annex with further details about the translation process and its correctness is available at https://github.com/qBioTurin/AB-ESSN.

Listing 1. Netlogo Code sample

```
;; declare main agent classes
breed [People a_People]
;; Agent attributes
People-own [Zones_0 Age_0 place myrate totrate rtime]
;; rtime variable has been added manually  to exploit agent-centered measures
```

Annex 1

The following section will be not included in final article in order to meet space requirements, but it will be available on the GitHub repository. We provide it here to give reviewers the opportunity to have an easier understanding of how the translation process takes place in practice.

Automatic Netlogo Translation. To demonstrate the feasibility of our translation approach, we implemented inside GreatSPN the automatic translation of the ESSN composed model into an Agent-Based Model that can be executed on NetLogo ABM software [44].

NetLogo is considered as one of the most easy programmable modeling environments for Agent-Based Models, and therefore it is often used as the preferred tool to introduce students into the world of ABM simulation. NetLogo considers two types of agents, "turtles" and "patches". The former best suit the common definition of agents; the latter are instead special agents representing the positions of the physical space where turtles may act and interact, thus patches cannot move.

Both can posses internal variables and be in charge of executing complex rules defined by the programmer through the "ask" command. Generally, the "setup" and "go" procedures are present inside any NetLogo program. The first procedure sets up the initial conditions, and the second one includes the simulation code that is continuously iterated until a given set of conditions defined by the user is reached. Even if the standard approach for time-advancement implemented in NetLogo is the Fixed Increment Time Advance (FITA) with equally spaced time-steps managed by means of a "tick" counter, the code produced by the translation algorithm implements a NETA (Next Event Time Advance) approach on top of that according to the simulation algorithm 1 defined earlier.

The translation starts with the declaration of all the defined agent classes as Netlogo turtles with the command "breed", and all the possible associated attributes with the command "<breed>-own".

For instance, on the SEIRS model in Figure 1, the code for the declaration of the agent class shown on Listing 1.

In the SEIRS example we have only one agent type (called *People*). Every agent will possess a *place* attribute, which encodes the ESSN place the agent is currently residing in (i.e., the agent state), a *myrate* list attribute to store the rates of the rules for which the agent will be the active leader, and *totrate* to store its cumulative rate. *myrate* and *totrate* are used to implement the NETA simulation algorithm we described in sec. 2.4. Furthermore, in this example agents also have two more custom attributes named *Zones_0* and *Age_0*. Such attributes, that translate the color classes *Zones* and *Age*, are used to represent agents' position and age range (i.e., young, mid, old), respectively. From the GreatSPN interface it is possible to declare the color classes associated with agents types by using the special keyword **agent** inside the class declaration (e.g., "**class** *People* = **agent** $pp\{1 \dots 100\}$" will define the ids for 100 agents of type People).

Furthermore, we note here that patches and Netlogo spatial representation are actually not considered for the moment. Their translation will be implemented in future releases.

A global variable is defined for each Place, and a unique numerical value is associated with it in the setup procedure. Then simulation parameters are set-up and the set of initial agents (corresponding to $\mathbf{m}_0$) is created, as shown in Listing 2:

Listing 2. Netlogo Code sample

```
set delta1 0.01
...
;; place identifiers
set SS 1000     ;;susceptible place identifier
set EE 1001     ;;exposed place identifier
set II 1002     ;;infected place identifier
set RR 1003     ;;recovered place identifier
...
;; setup initial marking
  create-People 90 [ ;; 90 people
    set place SS ;;state susceptible
    set Age_0 0 ;; young
    set Zones_0 0 ;; zone 0
    set color [100 149 237] ;; random color
  ]
  create-People 10 [ ;; 10 people
    set place II ;; state infected
    set Age_0 0 ;; young
    set Zones_0 1 ;; zone 1
    set color [100 149 237]
  ] ...
  let allAgents (turtle-set People )   ;;code added manually to inizialize
  ask allAgents [set rtime []  ]       ;;the  variable (treated as a list)
  ;;for storing reinfection times
```

A randomly-chosen color is associated with all the agents belonging to the same type. The agent features will be instead set according to the initial marking. Finally, static color subclasses of attribute color classes are created and allocated.

After such initial declarations, the annotated ESSN model is translated in NetLogo as well. Every ESSN transition t is associated with a *leading agent* chosen among the agents associated to the variables on the input arcs of t.

In the main loop (the *to go* procedure), each transition t is translated into an agent rule by using two separate code segments, as described in section 2.6. In the first code segment the rule rate for each agent is calculated. For simple transition rules (i.e., rules that involve only one agent) the rate calculation is trivial and it is directly obtained by the transition rate. For interaction rules (i.e., rules that involve more than one agent) each active agent needs to know the number of possible markings (i.e., sets of passive agents it can interact with) that satisfy the guard conditions, and then this number is used for the computation of the actual transition rate.This is calculated by means of nested loops on the

Listing 3. Netlogo Code sample

```
;; transition Infection
set _A1 People with [place = SS AND (True)] ;;Active leading agent
if any? _A1 [
  ask _A1 [
    let countInstances 0
    let p1 [who] of self        ;;saving agent identity
    let sus [Age_0] of self     ;;saving agent age
    let pos1 [Zones_0] of self  ;;saving agent position
    ;; Each agent in _A1 will select the set of compatible passive agents _A2
    let _A2 People with [place = II AND (pos1 = Zones_0)]
    if any? _A2 [
      ask _A2 [
        let p2 [who] of self
        let inf [Age_0] of self
        let pos2 [Zones_0] of self
        set countInstances  countInstances + 1
      ]
    ]
    ;; summing up all rates
    set myrate replace-item 9 myrate (countInstances * (infection))
  ]
]
```

participating passive agents, as shown in Listing 3, representing the translated code of the "infection" transition from the SEIRS example. [7]

Here, each active People agent with status *SS* (e.g., susceptible) will act as active leading agent by asking to People agents in ∈ *_A2* (e.g., passive agents in infected state on the same position of the leading agent) to count themselves (set countInstances countInstances + 1). In this way, active agents belonging to the set *_A1* will obtain the total number of combinations[8] of selected passive agents to calculate the total transition rate, considering only the agents that they can interact with. Actually, *myrate* is represented as an array of rates, and each element of the array stores the rate of a transition in which the agent is active.

After this step, that is executed for all the agents and all the rules for which the agents are active agents, each agent will be asked to calculate its cumulative rate inside their internal variable *myrate*, and then the total rate *totrate* is calculated as the sum of agents' rates. Thus, the selection of the next event starts by randomly choosing the next agent that will act, taking into account the weights of the event rates [9]. The example code for this process is reported in Listing 4.

After that, the second part of the code segment related to transitions is introduced. This code will include the effects related to the firing of the selected transition, and will be executed only by the chosen agent. Simply speaking, the next acting agent will select the next transition/rule (again with a roulette-wheel method) and will choose the passive agent(s) participating in the selected action. Finally, it will perform (or will ask to passive agents to perform) all agent feature updates according to the expressions on the transition arcs. This may lead to the following scenarios:

[7] The current implementation is in beta state and does not translate general laws for rates' calculations yet; these must be added manually.
[8] This mechanism for computing the number of combinations of passive agents that can interact with the leading agent can handle any number of passive agents and any transition guard constraining the allowed combinations, however in some cases, including the simple case illustrated in the example, it could be optimized by exploiting the agents counting functions of NetLogo.
[9] this result is achieved by using the NetLogo Rnd extension

Listing 4. Netlogo Code sample

```
let allAgents (turtle-set People )          ;; only  People angents can calculate their rate
ask allAgents [set totrate sum myrate]      ;; calculation of agents' total rate
set gammatot sum [totrate] of allAgents     ;; calculation of cumulative rate
if gammatot = 0 [stop]                      ;; stop if no more actions can occur
let increment ((-1 / gammatot) * ln(random-float 1)) ;; generate the random time advancement
;;according to  the exponential distribution and the total rate
set time  time + increment                  ;; increment time
;; select the next agent that will perform an action
let chosenAgent rnd:weighted-one-of allAgents [totrate]
```

- An agent variable that appears both on an input and an output arcs is updated (the place is changed, its attributes are modified according to the arc function);
- An agent variable that only appears on an output arc will lead to a newly created agent.
- An agent variable that appears in input but does not appear in output will lead to agent death.

For example, the *infection* transition will change the state of the leading active agent $p1$ to *EE* (e.g., exposed), but will not modify the state of the selected passive infected agent ($p2$). The example code for agent selection an infection transition/rule is reported in listing 5.

Once the event execution is completed, the same process is repeated by asking again all agents to update their transition/action rates and then to select the next agent/event.

The generated NetLogo model will include, besides the code, some basic features such as the setup and go buttons, the possibility to select a custom stop time for the simulation (default 0; no stop) and a "random seed" input field on the model interface to select the initial random seed of the simulation. Moreover, a basic real-time plot for the cumulative number of agents of the same type and sub-classes (i.e., number of agents of the same type but in a different states) is incorporated into the model interface.

In addition to the model translation, the instructions needed to compute the measures of interest are generated: in the examples discussed within the paper these are represented by the number of alive agents of each type. However, it is possible to enrich the model by introducing agent specific measures or observations for facilitating studies that require a micro-perspective, as in the case of the reinfection times reported for the SEIRS model. The code presented above also shows the instructions that were added manually to declare, initialize and update the $rtime$ variable, that was used for storing the reinfection times of each agent.

CORRECTNESS OF THE TRANSLATION

The generated ABM model is equivalent to the corresponding ESSN model it originated from as, by construction, a bijective relationship holds.

State bijection. Each ESSN subnet identifies a given agent type, the set of places within a given subnet identifies a state feature for that given agent type whose feature cardinality (i.e., number of possible values) is equal to the number of places in the subnet. The color domain of each place identifies the set of features of such agent type, including a color id that uniquely identifies tokens as agents (see for example the translated code in Listing 1). So, as we impose that each token must be unique (thanks to a given color id), each token will uniquely correspond to an agent. Thus, the ESSN state that is a given distribution of (colored) tokens in the places will be represented in the ABM as the union of

Listing 5. Netlogo Code sample

```
let bindingSelected false
  (if-else
  is-a_People? self [
    ;; select the next action performed by the chosenAgent
    let indices  n-values (length myrate) [ i -> i ]
    let pairs (map list indices myrate)
    let nextaction  first rnd:weighted-one-of-list pairs [ [var_P] -> last var_P ]
    (
      if-else
      nextaction = 0 [ ... ] ;; other actions not related to the infection transition/action
      ....
      nextaction = 4 [
        ;; chosenAgent is leader of Infection
        let targetRate random-float (item 4 myrate)
        let p1 [who] of self
        let sus [Age_0] of self
        let pos1 [Zones_0] of self
        let _AA2 People with [place = II AND (pos1 = Zones_0)]
        if any? _AA2 [
          ask _AA2 [
            if NOT bindingSelected [
              let p2 [who] of self
              let inf [Age_0] of self
              let pos2 [Zones_0] of self
              set targetRate targetRate - (infection)
              if-else targetRate > 0 [ ]
              [
                set bindingSelected true        ;; fire this binding
                ask turtle p1 [                 ;; agent p1 is modified
                  set place EE
                  set Age_0 sus
                  set Zones_0 pos1
                  set rtime lput time rtime    ;; code added manually to update
                  ;; the reinfection times list
                ]
                ask turtle p2 [                 ;; agent p2 is not modified
                ...

                ]
       ...          ;; here there are a series of closing brackets
      ]
       set place II
                   set Age_0 inf
                   set Zones_0 pos1
```

the states of all Agents. So there is a mapping from the ABM state and the Net and vice-versa. Note here that during translation the color id is not reported as in NetLogo Agents are already unique entities with their id.

Transtition/rule-instance bijection. The ESSN state change occurs in a given state for a given enabled transition instance. Correspondingly, In the ABM a state change occurs when a rule among the enabled ones is chosen and executed. In the ABM there is one rule for each transition, and each rule has an associated leader agent and possibly other agents

participating in the rule. The enabling of the rule in the ABM is checked by the leader agent, based on its state and the states of the other participating agents: in any state there is one enabled rule for each possible enabled instance of the corresponding transition and its rate is the same as that of the corresponding transition instance. The enabled instances of the different rules are determined by iterating on each leader agent whose state satisfies the requirement for rule enabling, and in case of an interaction transition involving two or more agents, this is followed by the nested selection of other agents satisfying the enabling conditions of the corresponding ESSN transition (see for example Listing 3).

The choice of the next rule application is done in a probabilistic manner driven by the rates according to a race conflict resolution policy and exploiting the properties of the exponential distribution (see for example Listing 4). The application of the rule (corresponding to the firing of a transition) performs the same type of state change as that defined by the transition, hence it will reach the same new state as the ESSN (see for example Listing 5).