S4. Mycobacteria

S4.1 Scope and schedule

S4 specifies individual mycobacteria and the containers that hold them. It covers the bacterium agent and its construction; containers and their location designations; biomass growth and asymmetric division; the modulation of growth by local caseation, crowding and adaptive immunity; the death routes; transfer and dispersal between containers; location re-designation; and the decay of dead bacteria. It expands the mycobacteria submodel of spine Element 7.

Bacteria are agents but are not scheduled

A bacterium derives from the common agent class and carries the common state (spine 2.5): an identifier, a birth time, a position and three receptor counts. In every other respect it is unlike the cellular agents.

It is not a grid occupant. Bacteria are held in containers: one per compartment for that compartment's extracellular population, and one per phagocyte for that cell's intracellular population (S4.3). A bacterium occupies no slot and is excluded from every occupancy predicate. Its position field keeps its default value and is never read; its location is that of the container holding it.

It is in no agent list, and the simulation holds no bacterial list. There is no bacterial movement pass, next-state pass or state-commitment pass. Most agent members do nothing, so a bacterium cannot be killed or suppressed through the agent interface (S4.2). Every change to bacterial state is made by a container on behalf of a caller.

Schedule

Bacterial processing occurs at several points of the cellular timestep (spine Element 3). Only the extracellular pass is a pass over bacteria; the others are calls made on containers by other submodels.

Block	Bacterial activity	Specified in
Next-state block, within a macrophage's rule	Uptake from the compartment container; intracellular growth immediately after uptake; intracellular killing by an activated macrophage; killing of extracellular bacteria at the macrophage's own compartment and, for an activated macrophage, at caseous neighbours	S2.5–S2.8; S4.4, S4.10, S4.12
Next-state block, on any macrophage death	Release, dispersal or destruction of the intracellular population, according to the death route	S2.9; S4.13
Extracellular pass, after all agent next-state computation and before daughter commitment	Per compartment: the radius-2 sweep giving the caseation count; extracellular growth and division; re-designation; density-dependent death in caseum; the healing test of S5; decay of the dead list	S4.4–S4.14
State-commitment and statistics blocks	Accumulation of the bacterial statistics	S9

Block	Bacterial activity	Specified in
Initialization	An empty container per compartment; the single intracellular bacterium of the seeding macrophage; any extracellular bacteria placed from the parameter file	Spine 5.4; S4.3

The extracellular pass visits compartments in row-major order. Within a compartment the order is: growth, re-designation and caseum death, then the healing test, then decay (S4.14).

A bacterium taken up during a macrophage's rule has left the compartment container before the pass reaches that compartment; the uptake rule therefore grows it intracellularly at once (S2.5). A bacterium released by a dying macrophage is in the compartment container before the pass arrives, and is grown extracellularly on the same timestep unless the pass has already visited that compartment. Whether released bacteria grow on the timestep of their release therefore depends on the position of the releasing compartment relative to the point the pass has reached.

The radius-2 sweep that supplies the caseation count also performs the entrapment test and its occupant kill (S5.4, S5.5).

Reads and writes

Reads: own biomass, stored growth rate, both intrinsic growth rates, generation time, division count, state and decay clock; the caseation status of the container's compartment; the caseation count (S5.5); the live count of the container; the current timestep and `Recruit_Prob_timeRecEnabled`; `Mtb_growthExtMtbBound` and the two computed caseation thresholds.

Writes: own biomass, stored growth rate, state, decay clock, generation time, birth time and division count; list membership; the daughter's identifier, parent identifier, birth time, biomass and generation time; the container's location designation; the extracellular containers of up to nine neighbouring compartments, by dispersal. The bacterial statistics are specified in S9.

Bacteria read no diffusing field. In particular they do not read oxygen: the hypoxia to which their growth responds is represented by the caseation count, not by the oxygen field, and the two are not coupled. They read no chemokine, TNF or IL-10, and no sink term of theirs depletes any field. They write no field either: a bacterium is not a secretion source, so the bacterial population influences the chemical environment only through the states it induces in macrophages and the cell deaths it causes.

S4.2 The bacterium agent

The state variables of a bacterium are inventoried in spine 2.5.

Constructors

The constructors used during a simulation differ only in the initial stored growth rate.

- **The counted constructor** sets the stored rate to the fixed rate it is given or, if growth-rate randomization is on (M8), to a uniform draw between the minimum and maximum it is given. It is used by the counted container constructor, the path by which every container holding bacteria at creation is built. An extracellular container passes the extracellular triple `Mtb_Agents_growthRateExtMtbMin`, `Mtb_Agents_growthRateExtMtbMax` and `Mtb_Agents_growthRateExtMtb`; a macrophage's intracellular

container passes the intracellular triple. The location argument determines which rate the dispatch reads (S4.5), not which triple was supplied.

- **The file constructor** sets the stored rate to 0. It is used only by the container's create-from-file path, the route by which extracellular bacteria are placed from the parameter file (spine 5.4).

Both set the intracellular intrinsic rate to `Mtb_Agents_growthRateIntMtb` and the extracellular intrinsic rate to `Mtb_Agents_growthRateExtMtb`, irrespective of where the bacterium will reside. Both set the biomass to `Mtb_Agents_initialBiomass` and the reserve to `Mtb_Agents_initialReserve`, and leave the generation time, division count, parent identifier and decay clock at 0.

Bacteria are also created by cloning: at division (S4.9) and when a container is deep-copied. The clone copies every field, then replaces the antibiotic concentration list with a deep copy of the original's. It makes no draw.

No senescence

Both constructors set the death time to -1. The next-state member does nothing, so the senescence test that every cellular agent evaluates is never reached and the sentinel is never compared. A bacterium dies only by the routes of S4.10.

Construction draws

Each agent built by a constructor, bacteria included, makes four draws on the real-number stream: three for the receptor counts and one log-normal draw for an NF-κB initial condition. Division daughters are copies and make none, but each division makes one draw for the asymmetric split (S4.9). In the default configuration the only constructed bacterium is the seeding one.

None of the four values is ever read. The receptor counts feed only coarse-grained occupancy proxies that a bacterium never invokes, and the NF-κB value feeds only the disabled pathway of M3. The draws nonetheless belong to the random stream.

Antibiotic concentration list

Each constructor fills the bacterium's antibiotic concentration list from the grid's antibiotic list. With M6 off no antibiotic exists, so the list is empty for every bacterium and every loop over it does nothing (S4.15).

Members that do nothing

The members for movement, secretion, next-state computation, state commitment, statistics, degradation, killing and suppression have empty bodies. Bacteria therefore do not move; they are relocated only by transfer or dispersal (S4.12, S4.13). They are not a source for any field and apply no receptor sink. The pending state is never committed by the agent, both states being written together at death (S4.10). Bacterial statistics are accumulated by the container, not the agent (S9).

The caseation cascade iterates the agent slots of a compartment and calls the killing member on each occupant (S5.3); the Treg suppression sweep (S3) iterates the same slots and calls the suppression member. Neither reaches a bacterium, because bacteria occupy no slot and because the members would do nothing if they did. Extracellular bacteria in a newly caseous compartment are not killed by the cascade; they die, if at all, by the density-dependent rule of S4.11.

S4.3 Containers and the location designations

The two lists

A container holds a live list and a dead list. It owns its bacteria and deletes both lists when it is destroyed. Ordinary accessors report on the live list alone; the dead list is reached only by explicit reference to the dead state. The accessors other rules use are:

- the live count and the dead count;
- the biomass sum, the sum of biomass over the live list;
- the mean growth rate, the mean stored rate over the live list, or 0 for an empty list;
- the reserve sum over the live list ("Reserve variables").

The biomass sum and the live count are different quantities, and one rule distinguishes them: a resting macrophage compares the biomass sum, not the count, with its uptake amount (S2.5). A bacterium's biomass lies in [0.86, 2) (S4.8).

Bacteria in one container share an environment, so after a growth pass their stored rates are equal and the mean growth rate is that common value. Bacteria that have just arrived by uptake, transfer or dispersal carry the stored rate of their previous environment until the next growth pass overwrites it. In that interval the mean is a mixture.

The location designations

Every container carries one of three location designations: intracellular, replicating extracellular or non-replicating extracellular. It is set at construction, changed by re-designation (S4.14), and passed to each bacterium on every growth step.

In the default configuration the designation has one live consumer, the growth-rate dispatch (S4.5). The dispatch distinguishes intracellular from extracellular but not the two extracellular values from each other. The replicating and non-replicating designations therefore affect no trajectory. Bacteria in a caseous compartment do not grow because the caseation modulator returns zero there (S4.6), not because their container is labelled non-replicating. Bacteria whose rate has fallen to zero through crowding or the caseation count are labelled non-replicating, and the label changes nothing about them. The designation is a reporting classification that does not feed back.

Its other consumers belong to M6 and are reached with an empty antibiotic list (M6.11). The statistic reporting the non-replicating population tests for a zero mean growth rate, not for the designation (S9).

Creation and ownership

Container	Created by	Count at creation	Designation at creation
One per compartment	The grid constructor, before any agent exists	Zero	Replicating, unless the compartment is already caseated, then non-replicating

Container	Created by	Count at creation	Designation at creation
One per macrophage	The macrophage constructor	The macrophage's initial intracellular load	Intracellular

Extracellular bacteria specified in the parameter file are added later in initialization, by the create-from-file path (spine 5.4).

A macrophage's container is created, cloned and deleted with the macrophage. When a macrophage is removed, its container and every bacterium left in it are destroyed. Every macrophage death route must therefore release, disperse or kill the intracellular population before the cell is removed (S4.10).

The compartment containers are shared between the current and next grids rather than duplicated. A change to a container is visible at once through either grid, and bacterial state is not subject to the current-versus-next discipline that governs the cytokine fields. A reimplementation must not duplicate the containers when it duplicates the fields.

Reserve variables

Five variables are declared and read by no rule: the reserve, reserve production, reserve consumption, biomass degradation rate and effective growth rate.

S4.4 The growth pass

Growth is a container operation, not an agent operation. The container shuffles its live list, grows each bacterium in turn (S4.8), withholds the daughters produced, and appends them after the loop. No bacterium grows twice in one pass, and a daughter never grows on the timestep of its creation.

Entry points

Entry point	Called from	Caseation flag	Caseation count
Intracellular growth	The uptake branch of the resting rule (S2.5); the infected (S2.6) and chronically infected (S2.7) rules; the suppressed branch (S2.3)	False, hard-coded	0, hard-coded
Extracellular growth	The extracellular pass, once per compartment holding at least one live bacterium	The compartment's caseation status	The count from the radius-2 sweep (S5.5)

The dispatch ignores the hard-coded arguments for an intracellular container (S4.5). An intracellular population is therefore insensitive to caseation and crowding, and its only modulation is the adaptive-immunity factor (S4.7). A macrophage cannot occupy a caseous compartment (spine 2.3), so the caseation flag could not be true. The crowding term is not applied either, so intracellular growth is unbounded by local density.

A chronically infected macrophage's container always holds bacteria. An infected macrophage's container may be empty (S2.6), and its growth pass then consumes no draws.

The pass

```
grow container(caseated, c):
    shuffle the live list
    daughters <- empty
    starved <- 0
    for each bacterium b of the live list, in order:
        d <- grow b(t, starved, caseated, c,
                    n = current size of the live list,   // re-read every iteration
                    randomization flag, designation,
                    scaling factor)                      // S4.8
        if b is dead:                                    // starvation only
            move b to the dead list and continue with the next bacterium
        else if d exists:
            append d to daughters
    add starved to the starvation statistic
    add the size of daughters to the division counter
    append daughters to the live list
```

The group size n is the current size of the live list, including the bacterium itself, re-read on every iteration. It could change during a pass only if a bacterium were erased or a daughter appended. Daughters are withheld, and erasure follows only a starvation death, which is unreachable (S4.8). The group size is therefore constant across a pass, by consequence rather than by construction.

The erase branch is never taken in the default configuration. It is nonetheless live code: any parameterization that lets biomass fall would reach it.

The two-dimensional to three-dimensional scaling factor computed by the observation layer on the previous timestep is forwarded to each bacterium. Only the M6 mutation loop uses it (S4.15).

Effect of the shuffle

The rationale for the shuffle is that each bacterium's growth consumes local nutrients. Nutrient limitation is not represented. The quantity that could make order matter within a pass is the group size, which enters the crowding term (S4.6). Were a bacterium able to die mid-pass, the bacteria processed after it would see a smaller group, and so a larger crowding factor, and the shuffle would decide which bacteria those were. Because the group size is constant, the shuffle affects no aggregate quantity. It still consumes draws and decides which bacterium receives which asymmetry draw at division.

S4.5 Growth-rate dispatch

Each bacterium computes its effective growth rate at the start of its growth step. It uses its location designation, its rates, the current timestep and the two environmental quantities its container forwarded. It writes the rate back into itself, then applies it to its biomass.

```
growth rate(designation, randomize, t, caseated, c, n):
    // Stage 1: select the unmodulated rate
    if randomize:                                    [M8]
        r <- stored rate
    else if designation is intracellular:
```

```
        r <- intrinsic intracellular rate
    else:
        r <- intrinsic extracellular rate
    // Stage 2: modulate by environment
    if designation is intracellular:
        r <- adaptive-immunity modulator(t, r)        // S4.7
    else:
        r <- caseation modulator(r, caseated, c, n)   // S4.6
    return r
```

The two stages are independent conditionals, so the randomization branch of stage 1 does not suppress stage 2: under M8 a randomized intracellular bacterium takes its stored rate and then receives the adaptive-immunity factor. In the default configuration the randomization branch is not taken, and stage 1 reduces to the location selection. Stage 2 distinguishes only intracellular from extracellular. The dispatch makes no draw on any path.

The three growth-rate fields

Field	Set at construction	Written during a simulation by	Read by
Intrinsic intracellular rate	`Mtb_Agents_growthRateIntMtb`	The M6 fitness cost and compensation only	Stage 1, intracellular
Intrinsic extracellular rate	`Mtb_Agents_growthRateExtMtb`	The M6 fitness cost and compensation only	Stage 1, extracellular
Stored rate	The fixed rate, or a draw under M8 (counted constructor); 0 (file constructor)	The write-back on every growth step; the copy into a daughter at division	Stage 1 under M8 only; the mean growth rate (S4.3); the growth-rate histogram (S9)

The intrinsic rates are invariant by default and identical for every bacterium, both being set from parameters that are not sampled. Every bacterium carries both wherever it resides, so an intracellular bacterium holds an extracellular rate it will not read unless it is released. The stored rate is not an intrinsic property: it records the most recent environmental outcome.

Write-back

Immediately after the dispatch, and before the rate is applied, the bacterium stores it: $\text{stored rate} \leftarrow r$, then $\text{biomass} \leftarrow \text{biomass} + r \times \text{biomass}$ (S4.8). After its first growth step, a bacterium's stored rate is therefore the fully modulated rate it applied on its latest step. This makes the mean growth rate a report of local conditions, and makes the growth-rate histogram a distribution of realized rates.

A bacterium built by the file constructor holds a stored rate of 0 until its first growth step, so it would contribute a zero to the growth-rate histogram and the mean growth rate on its creation timestep. No bacterium is created by that path by default, so the zero contribution does not arise. A bacterium built by the counted constructor starts with the fixed rate.

A daughter has no interval at zero either. After a division the parent copies its own stored rate into the daughter. The copy repeats the clone (S4.9), which has already copied the field, and both carry the same value because the write-back precedes the division test. It is the only place where one bacterium's stored rate is written by another, and under M8 it is the mechanism by which a lineage inherits a growth rate (M8.7).

A bacterium that arrives by uptake, transfer or dispersal carries the stored rate of its previous environment until the next growth pass. For an uptaken bacterium the interval is nil, since the uptake rule grows it intracellularly at once (S2.5). For a dispersed bacterium it depends on the order of the extracellular pass (S4.1).

The intrinsic rates are inherited by the clone and are never redrawn or reset. A lineage is therefore uniform in its intrinsic rates, and would diverge only through the resistance mutations of M6.

S4.6 Caseation and crowding modulation of extracellular growth

The extracellular modulator is a pure function of its four arguments. It applies an absolute arrest in caseous compartments, a multiplicative crowding term in the local live count, and a three-branch reduction in the caseation count.

```
caseation modulator(r, caseated, c, n):
    if caseated:
        return 0
    B <- Mtb_growthExtMtbBound                          // 114.527
    a <- 0
    if n < B:
        a <- 1 - n / B
    if c < Mtb_Agents_growthRateExtMtbCaseated:          // c < 5
        return r * a
    else if c < Mtb_Agents_noGrowthExtMtbCaseated:       // 5 <= c <= 13
        return (r / c) * a
    else:                                                // c >= 14
        return 0
```

The caseous early return precedes everything. A bacterium in a caseous compartment has an effective rate of exactly 0, whatever its crowding term and caseation count. This, and not the container's designation, is what stops growth in caseum.

The crowding term

The group size n is the size of the container's live list when the bacterium is processed, including the bacterium itself, so a lone bacterium passes 1. With B = `Mtb_growthExtMtbBound`, the term is $a(n) = 1 - n/B$ for n < B and 0 for $n \geq B$.

The clamp at zero makes every effective rate non-negative. Without it the rate would be negative for n > B, biomass would fall, and the starvation route of S4.8 would become reachable; this is the rationale for the clamp.

B is not an integer, so the strict comparison admits n = 114, with a = 0.0046, and excludes n = 115, with a = 0. A compartment holds at most 114 growing bacteria; one holding 115 or more is static unless its population falls.

The term is linear, not saturating, so it bites well below the bound: a compartment holding 10 bacteria grows at 91 % of the unmodulated rate, and one holding 50 at 56 %.

The caseation count

The caseation count c is produced by the radius-2 sweep of S5.5. It is the number of compartments of the periodic Moore neighbourhood of radius 2, 25 in all, that are caseous, plus the number that hold a chronically infected macrophage. c is at most 25, except transiently where a senescing chronically infected macrophage remains in a newly caseous compartment; the formal ceiling is 50. Within one timestep, entrapment changes the counts seen by compartments swept after it, so c depends on the sweep order (S5.5).

A compartment holding a chronically infected macrophage counts toward c, and so toward both thresholds (S4.16), equally with a caseous compartment. A bacterium surrounded by chronically infected macrophages in uncaseated tissue meets the same reduction as one surrounded by caseum. Five such macrophages within radius 2 cross the lower threshold and fourteen arrest growth, so a dense core of chronically infected macrophages arrests extracellular growth in its neighbourhood in the absence of caseation.

The branch boundaries fall at 20 % and 56 % of the neighbourhood. The upper sits at its fraction of 56 %. The lower sits below its fraction of 23 %, because the floor in the threshold derivation truncates 5.75 to 5 (S4.16).

The middle branch

For $5 \leq c \leq 13$ the rate is divided by the count itself, not interpolated between the thresholds, so the reduction factor is 1/c. The function is discontinuous at both ends of the branch. Moving from c = 4 to c = 5 reduces the rate fivefold; moving from c = 13 to c = 14 takes it from 1/13, or 7.7 %, to zero. Within the branch the factor varies only 2.6-fold across nine integer values. The branch therefore behaves less like a graded transition than like a fixed reduction of about an order of magnitude with a weak dependence on the count.

Consequences

With `Mtb_Agents_growthRateExtMtb` = 0.0025, a bacterium at the crowding bound, in a neighbourhood with $c \geq 14$, or in caseum does not divide within a run of 22176 timesteps. A bacterium in the middle branch still doubles within about 10 to 25 days. Containment of the extracellular population is thus achieved by arrest rather than by killing, produced jointly by local crowding and the caseation count. The modulator makes no draw.

S4.7 Adaptive-immunity modulation of intracellular growth

The intracellular modulator applies one multiplicative factor after a fixed onset, and is otherwise the identity.

```
adaptive-immunity modulator(t, r):
    if t >= Recruit_Prob_timeRecEnabled + Mtb_Agents_growthRateFactorDelayIntMtb:
        r <- r * Mtb_Agents_growthRateFactorPostAdaptiveIntMtb
    return r
```

The comparison is non-strict. The rationale is that intracellular bacteria slow their growth as the immune system activates, as a proxy for a response to iNOS and arginase induction that the model does not represent.

The factor applies wherever intracellular growth is performed (S4.4). The test reads only the timestep. It does not depend on whether any T cell is present, on bacterial load, or on the state of the host macrophage.

With `Mtb_Agents_growthRateFactorDelayIntMtb` = 0 and `Mtb_Agents_growthRateFactorPostAdaptiveIntMtb` = 1, the modulator returns its argument unchanged on both branches. It is live code with an effect of nil: a factor below 1 would make it operative with no other change.

The onset test reads the parameter `Recruit_Prob_timeRecEnabled`, not the per-simulation onset that the simulation draws and stores. With `Recruit_Prob_timeRecEnabledRange` = 0 the drawn onset equals the parameter, and the modulator switches one timestep before T cell recruitment effectively starts (spine 5.7). The mechanism is inert by default.

When `Recruit_Prob_mtbRecEnabled` > 0, recruitment begins when total bacterial load first exceeds a threshold, and the factor still begins at `Recruit_Prob_timeRecEnabled` plus the delay.

The delay is declared as a real number with no range, and the test compares an integer timestep against a real sum. A non-integer delay therefore takes effect at the first timestep at or above the sum, so it is effectively rounded up. A negative delay brings the onset forward, before `Recruit_Prob_timeRecEnabled`.

The factor is declared on [0, 1], so it can slow intracellular growth but not accelerate it. At 1, its default value, multiplication leaves every rate bitwise unchanged (M8.2). For 0 < f < 1 with growth-rate randomization off, each step's intracellular rate after onset is $f \times$ `Mtb_Agents_growthRateIntMtb`; with it on, the rates compound as f^m over m steps (M8.7).

With f = 0, intracellular bacteria no longer divide after onset. An infected macrophage can still become chronically infected by uptake, which slows as its load nears the threshold. A chronically infected macrophage takes up nothing (S2.7), so it holds its load and never bursts.

Consequences for intracellular growth

This is the only modulator applied to an intracellular container (S4.5), and it is the identity by default, so intracellular growth is unmodulated. Every intracellular bacterium grows at `Mtb_Agents_growthRateIntMtb` = 0.0045 on every timestep, a biomass doubling time of $\lceil \ln 2 / \ln 1.0045 \rceil = 155$ timesteps, or 25.8 h.

- **No crowding or caseation term** (S4.4). A macrophage's population grows at the same per-bacterium rate whether it carries one bacterium or forty, so it grows exponentially until bursting removes the host (S2.7). The burst threshold is thus the only quantity bounding intracellular residence and amplification.
- **No response to local chronic infection.** A bacterium inside a macrophage in the densest part of a chronically infected core grows at the full intrinsic rate, while a bacterium one compartment outside it in the extracellular space does not grow at all (S4.6).

The intrinsic rates stand in the ratio 0.0045 : 0.0025 = 1.8. The realized intracellular advantage exceeds that ratio wherever the extracellular environment is crowded or has a high caseation count.

S4.8 Biomass, division and starvation

Growth is applied to one scalar, the biomass, and its two outcomes are tested against two fixed thresholds on that scalar.

```
grow b(t, starved, caseated, c, n, randomize, designation, scaling factor):
    r <- growth rate(designation, randomize, t, caseated, c, n)   // S4.5
    stored rate <- r
    biomass <- biomass + r * biomass
    daughter <- none
    if biomass >= Mtb_Agents_divisionThreshold:                   // >= 2
        daughter <- divide(t, designation, scaling factor)         // S4.9
        daughter's stored rate <- stored rate
    else if biomass < Mtb_Agents_deathThreshold:                  // < 0.5
        die()                                                     // S4.10
        starved <- starved + 1
    return daughter
```

The division test comes first and starvation is its alternative branch, so at most one fires on a given step. The update $b \leftarrow b(1+r)$ makes r a fractional rate per cellular timestep, not an absolute increment. The container interface uses this convention throughout; it admits a negative rate (S4.6). No rule other than growth and division writes the biomass.

Doubling times

An unmodulated bacterium's biomass doubles in $\lceil \ln 2 / \ln(1+r) \rceil$ timesteps: 155 timesteps (25.8 h) at the intracellular rate of 0.0045, and 278 timesteps (46.3 h) at the extracellular rate of 0.0025. The interval between successive divisions of one bacterium differs, because division is asymmetric (S4.9). At the intracellular rate, a product receiving the larger share, $0.57 \times 2 = 1.14$, reaches the threshold in 126 timesteps; one receiving the smaller share, 0.86, needs 188. The population doubling time is the biomass doubling time, and individual generation times straddle it.

Starvation is unreachable

- **Every effective rate is non-negative.** The intracellular path returns the intrinsic rate times a factor of 1 (S4.7). The extracellular path returns 0, or the intrinsic rate times the crowding term, or the intrinsic rate divided by a positive integer and times the crowding term; the crowding term is clamped at zero from below (S4.6).
- **Biomass is therefore non-decreasing except at division.** With $r \geq 0$, $b(1+r) \geq b$.
- **Every attainable biomass exceeds the threshold.** A bacterium starts with biomass `Mtb_Agents_initialBiomass` = 1 if built by a constructor, or $f \times b$ at division, where $b \geq 2$ is the parent's pre-division biomass and f is 0.43 or 0.57. The smallest attainable biomass is $0.43 \times 2 = 0.86$, reached when a bacterium divides at exactly the threshold and takes the smaller share. Both 1 and 0.86 exceed `Mtb_Agents_deathThreshold` = 0.5, and biomass only increases thereafter, so the test b < 0.5 is never satisfied.

More generally, starvation cannot occur while rates are non-negative and `Mtb_Agents_deathThreshold` is no greater than the smaller of `Mtb_Agents_initialBiomass` and $\min(\phi, 1-\phi)\times$ `Mtb_Agents_divisionThreshold`, with $\phi =$ `Mtb_Agents_divisionFraction`. The default values satisfy this with a wide margin. Starvation would need a division fraction below 0.25, a death threshold above 0.86, or a negative effective rate.

It follows that the starvation statistic is identically zero, that the erase branch of the growth pass is never taken, and that the group size is constant across a pass, which is what makes the shuffle inconsequential (S4.4). The reachable bacterial death routes are those marked reachable in S4.10.

S4.9 Asymmetric division

A bacterium whose biomass has reached the division threshold divides into two bacteria whose biomasses sum to the parent's pre-division biomass but are unequal. One draw decides which product receives the larger share.

```
divide(t, designation, scaling factor):
    if U(0,1) < 0.5:
        daughter fraction <- Mtb_Agents_divisionFraction          // 0.43
        parent fraction   <- 1 - Mtb_Agents_divisionFraction      // 0.57
    else:
        daughter fraction <- 1 - Mtb_Agents_divisionFraction
        parent fraction   <- Mtb_Agents_divisionFraction
    daughter <- clone()                                 // copy; no draw
    update(t, parent fraction)                         // on the parent
    division count <- division count + 1
    daughter: update daughter(t, daughter fraction, identifier, generation time)
    M6 mutation steps                                   [M6]  (S4.15)
    return daughter

update(t, f):
    biomass <- biomass * f
    reserve <- reserve * f
    generation time <- t - birth time
    birth time <- t

update daughter(t, f, parent identifier, parent generation time):
    identifier <- next value of the global counter
    parent identifier <- parent identifier
    division count <- 0
    birth time <- t
    update(t, f)
    generation time <- parent generation time
```

The coin is fair and is drawn afresh at every division, so the asymmetry has no persistent direction: a parent is as likely to keep the smaller share as the larger, and no lineage accumulates an advantage. The parameter sets the magnitude of the asymmetry, not its sign. The two products receive 0.43b and 0.57b.

Because the parent's update has already run, the generation time passed to the daughter is the parent's just-completed generation time. Both products therefore report the same generation time immediately after a division: the interval the parent took to reach the threshold. The reserve is scaled with the biomass, with no effect, since it is identically zero (S4.3).

Fields of the daughter

Field	Value in the daughter
Identifier	Next value of the global counter
Parent identifier	The parent's identifier
Division count	0
Birth time	t
Biomass	The parent's pre-division biomass times the daughter's fraction
Generation time	The parent's just-completed generation time
Stored rate	The parent's, by the copy and by the explicit copy of S4.5
Everything else	Inherited unchanged by the copy

Everything else includes the death time of -1, both intrinsic rates, the state, the decay clock, the receptor counts and the NF-κB value. The copy makes no draw, so the daughter does not make the construction draws of S4.2.

Lineage depth is unbounded

The division count is incremented on the parent and reset to 0 on the daughter. It is therefore a count of divisions performed by one bacterium, not a depth within a lineage. No threshold is tested against it, and nothing bounds the number of generations a lineage may produce; the count is an observation quantity only. This is the opposite convention from T cells, whose counter is inherited by the daughter and capped (S3).

S4.10 Death routes

Death and destruction are distinct and are accounted differently.

The state change

A bacterial death writes the current and pending states to dead together and sets the decay clock to `Mtb_Agents_decayTime` (5 timesteps). A bacterial death is therefore immediate; it never passes through the next-state mechanism, and there is no pending-death state.

The two container routines

```
die at position(p):                        // positional
    move the bacterium at p to the dead list and mark it dead
    return the next valid position in the live list

die count(k):                              // counted; requires k <= live count
    if k < live count:
        shuffle the live list
    mark the last k bacteria of the live list dead and move them to the dead list
```

The positional form is used by routines already iterating the live list: the starvation branch of the growth pass (unreachable, S4.8), caseum death (S4.11) and the drug-killing routines of M6. It returns the next valid position, so a caller can keep traversing a list it is modifying.

The counted form is used by routines that kill a quantity without regard to which bacteria: macrophage killing of extracellular and intracellular populations, and the clean cytotoxic kill. It shuffles the live list and takes from its end, so the selection is uniform. The shuffle is skipped when the count equals the whole list, since the selection is then vacuous.

The counted form has wrappers. Kill-all kills the whole live list and returns its size. Kill-count truncates its argument to an integer and delegates to kill-all if the count equals or exceeds the live count, otherwise killing the count. It therefore never kills more bacteria than exist, and no caller need check. Both return the number killed.

Routes

Route	Called by	Routine	Reachable
Starvation	The bacterium's own growth routine	Positional	No (S4.8)
Uptake-killing by a resting macrophage	S2.5	Kill-all if the local biomass sum is below the uptake amount, else kill-count with the uptake amount	Yes
Killing by an activated macrophage, own compartment	S2.8	Kill-count with `Core_Mac_nrExtMtbUptakeAct`	Yes
Killing by an activated macrophage, caseous neighbours	S2.8	Kill-count with `Core_Mac_nrExtMtbUptakeAct` , on a draw per caseous neighbour (S2.8)	Yes

Route	Called by	Routine	Reachable
Intracellular killing by an activated macrophage	S2.8	Kill-count with 1, one bacterium per timestep	Yes
Density-dependent death in caseum	The extracellular pass	Positional, per victim (S4.11)	Yes
Clean cytotoxic kill	S3	Kill-all, through the macrophage's clean-kill routine	Yes
Drug killing	M6	Positional	No (S4.15)

For the caseation cascade, see S4.2. A T cell never kills a bacterium directly: both T cell killing routes act on the macrophage, and the bacterial outcome is whatever the macrophage's death route does with its container (S4.13).

Destruction without death

Destruction removes bacteria without the state change, by the paths tabulated. The bacteria are deleted outright, so they never acquire the dead state, never enter a dead list and never decay.

Path	What is destroyed	Counted as
The residue of a dispersal	Every bacterium of the dispersed list not allocated to a compartment (S4.13)	Killed, through the dispersal routine's return value
Container destruction	Everything left in a macrophage's container when the macrophage is removed	Nothing

Container destruction follows the macrophage death routes, which have already emptied the live list by releasing, dispersing or killing it, so in normal operation it destroys only dead bacteria. In a clean cytotoxic kill, kill-all moves the whole live population to the dead list, and the container is destroyed with the macrophage at the end of the timestep. Those bacteria are counted as killed but are deleted before their decay clocks are read, so a clean kill leaves no detectable bacterial remnant at all.

The distinction matters because the dead list is detectable. A macrophage's exposure memory is set by live or dead extracellular bacteria in its neighbourhood (S2.10). A death that places a bacterium in a compartment's dead list leaves it there for five timesteps (S4.14); a destruction leaves nothing. The fate of a macrophage's dead list on each death route is given in S2.9, and the mortality statistics in S9.

S4.11 Density-dependent death in caseum

Bacteria in a caseous compartment do not grow (S4.6). Without a removal mechanism, a caseous compartment's extracellular population would be fixed at whatever had accumulated there and would rise as caseation spread. A density-dependent death rule therefore applies to caseous compartments only, once per cellular timestep.

The extracellular pass calls the rule when the compartment is caseous and the M6 guard against caseum death is false (M6.14); the guard is false by default. The number killed is added to the caseum bacterial-kill statistic. That statistic is also credited with the dispersal destruction tolls of macrophages killed by the caseation cascade and by entrapment, but those tolls are zero at a dispersal fraction of 1.0 (S4.13), so it counts caseum deaths only (S9).

```
caseum death():
    killed <- 0
    p <- min(exp((live count - Mtb_growthExtMtbBound) / 10), 1)
    for each bacterium of the live list, in list order:
        if U(0,1) < p:
            die at position
            killed <- killed + 1
            p <- min(exp((live count - Mtb_growthExtMtbBound) / 10), 1)
    return killed
```

Every bacterium in the live list at entry is visited exactly once, since the positional routine returns the next valid position, and each takes one draw whatever the outcome. The probability is recomputed after each death, so the death probability falls within a pass: later bacteria face a probability lower by a factor of $e^{-0.1}$ per death already effected. The list is not shuffled first; its order is that left by the growth pass. The divisor 10 is fixed in the source. The bound B = `Mtb_growthExtMtbBound` is shared with the crowding term (S4.6) and with the exceedance statistic (S9). The subtraction is performed in floating point, so the difference is negative for counts below the bound.

Magnitude

The probability $p(n) = \min(\exp[(n - B)/10], 1)$ is negligible below about n = 70 and severe above about n = 100. The rule therefore imposes a ceiling rather than clearing a compartment: a pass beginning at n = 115 kills about 25 to 30 bacteria, leaving the compartment near 85 to 90, where the per-timestep probability has fallen to a few per cent. Since growth in caseum is zero, the population can rise below that level only by influx from dispersal, release or bursting nearby. The ceiling lies below the growth bound, so a caseous compartment holds fewer bacteria than a crowded non-caseous one.

A compartment holding a single bacterium kills it with probability 1.17×10^{-5} per timestep. Over the 22176 timesteps of a run the probability that it is ever killed is $1 - (1 - 1.17 \times 10^{-5})^{22176} = 0.23$. A lone bacterium in caseum therefore survives the whole run with probability about 0.77, neither growing nor dying, and is invisible to every immune mechanism except the activated-macrophage sweep of S2.8, which reaches caseous compartments from outside.

S4.12 Transfer

Bacteria move between containers by transfer. One routine performs every movement, and wrappers expose it for uptake into a phagocyte, release onto the grid, partial release, and the per-compartment allocations of a dispersal (S4.13).

```
transfer(source list, destination list, k):
    shuffle the source list
    move the last k bacteria of the source list to the end of the destination list
```

The source list is shuffled and the bacteria are taken from its end, so the selection is uniform over the source population. The destination receives them appended, so its order reflects arrival until its next shuffle. The shuffle is performed even when the whole source list moves, unlike the counted death routine (S4.10).

By default no caller requests more bacteria than the source holds. Uptake requests one bacterium and is guarded on a non-empty source; full release requests exactly each list's size; each dispersal allocation is taken from a source already reduced by the earlier allocations, and the allocations sum to at most the list size; partial release has no caller.

Wrapper	Lists	Semantics
Uptake	Live list only	Moves the count from a compartment container to a phagocyte's container
Full release	Dead list, then live list	Moves each list whole to the destination
Partial release	One named list	Moves the count, then destroys the rest of that list; no caller
Dispersal allocation	One named list	Moves an allocation to one compartment (S4.13)

Uptake moves live bacteria only, so a phagocyte never takes up a dead bacterium. Dead extracellular bacteria stay in the compartment's dead list, keep decaying there, and keep setting the exposure memory of nearby macrophages until deleted (S2.10). A macrophage can therefore be exposed by bacteria it cannot take up. The uptake amount and the guards on the two uptake paths are specified in S2.5 and S2.6.

The create-from-file path creates new bacteria through the file constructor (S4.2); it is not part of the transfer interface.

S4.13 Survival of Mtb after macrophage death, and dispersal

Survival of Mtb after macrophage death

A macrophage's death route decides what happens to its bacteria; the route-by-route fate and the dispersal fraction of each route are tabulated in S2.9.

- **Local release, on senescence.** The whole container goes to the extracellular container of the macrophage's own compartment by full release (S4.12): the dead list first, then the live list. Senescence is the only route that keeps the whole intracellular population, live and dead, at the macrophage's own compartment. Since no rule depends on a bacterium's position within a list, the order affects no aggregate quantity.
- **Dispersal.** Every other route except the clean cytotoxic kill disperses the container, with a fraction of 0.5 or 1.0.
- **Kill in place.** The clean cytotoxic kill kills the whole live list, and the container is destroyed with the cell (S4.10).

Dispersal

Dispersal relocates a fraction of a container's bacteria into the extracellular containers of the Moore neighbourhood and destroys the rest. The live list and the dead list are dispersed with the same fraction, by two separate calls of the per-list routine. Each call builds and shuffles its own permutation of compartments, so the live and dead bacteria of one macrophage go to different random subsets of the neighbourhood. Only the live list's destruction toll is returned; the dead list's is discarded.

```
disperse list(list, f):
    n <- size of the list
    d <- floor(f * n)
    if f = 0 or d < 1:
        destroy every bacterium of the list
        return n
    permutation <- the ordinals of the Moore neighbourhood, shuffled
    k <- min(d, size of the Moore neighbourhood)            // at most 9
    for i from 0 to k - 1:
        allocation <- floor(d / k) + (1 if i < d mod k else 0)
        transfer allocation bacteria to the compartment at permutation[i]   // S4.12
    destroyed <- number of bacteria left in the list
    destroy every bacterium left in the list
    return destroyed
```

For $d \leq 9$ the compartment count equals d, so each of d compartments receives one bacterium. For d > 9 all nine compartments are used, the remainder going one each to the first d mod k entries of the shuffled permutation. Every compartment, the centre included, receives at least one bacterium whenever nine or more are dispersed.

The residue left after the allocations is destroyed, not killed (S4.10). A fraction of 1 destroys nothing, since the truncation is then exact. At 0.5, truncation biases dispersal toward destruction when the burden is odd, so "half dispersed" is exact only for even burdens.

n	f	Dispersed	Destroyed	Allocation
1	0.5	0	1	Early return
3	0.5	1	2	1 to one compartment
20	0.5	10	10	2 to one compartment, 1 to each of eight
41	1.0	41	0	5 to each of five compartments, 4 to each of four

A single bacterium is destroyed by half-dispersal

An infected macrophage may carry exactly one bacterium. For n = 1 at a fraction of 0.5 the truncated quantity is 0, so the early return destroys the bacterium rather than relocating it. TNF-induced apoptosis and Fas-mediated killing of a macrophage carrying one bacterium therefore eliminate that bacterium. A newly infected macrophage may hold a single bacterium for up to about a day; during that time apoptosis and Fas killing are sterilizing.

Targets ignore caseation and occupancy

The destination is the extracellular container of the drawn compartment, taken with no predicate.

- **Caseation is not tested,** so dispersal places bacteria in caseous compartments freely. This feeds the arrested caseous population of S4.11, and contrasts with cells, which can neither enter nor remain in a caseous compartment (spine 2.3).
- **Occupancy is not tested,** and compartments have no bacterial capacity. The slot limits apply to cells only. A compartment may hold any number of bacteria, and the only density effects are the crowding term (S4.6) and caseum death (S4.11). A non-caseous compartment can therefore hold an arbitrarily large static population.
- **The centre compartment is included,** so a dispersal may return bacteria to the dying macrophage's own compartment.
- **Targets wrap periodically,** so a dispersal at a domain margin places bacteria at the opposite margin, as agent sensing does (spine 2.2).

The routine returns the destruction toll only. The caller computes the quantity released, the pre-dispersal live count less the toll, for the outside-granuloma accounting (S9). Nothing records the compartments dispersed to or the dead list's toll.

S4.14 Re-designation and decay

Re-designation and decay close each compartment's work in the extracellular pass. Re-designation relabels the container according to whether its bacteria are growing. Decay advances the clock of every dead bacterium and deletes those whose clock has run out.

Re-designation

Immediately after the growth call, the pass reads the container's mean growth rate m and applies the demotion test, then the promotion test:

```
// demotion test
if designation is replicating and (m = 0 or caseous):
    designation <- non-replicating
// promotion test
if designation is non-replicating and m != 0 and not caseous:
    designation <- replicating
```

The promotion test reads the designation afresh, so a container just demoted is re-examined. The two conditions are mutually exclusive, so no container flips twice in one timestep.

Immediately after the growth pass, every live bacterium in the container carries the same effective rate, a function of the container's environment alone (S4.5), and daughters inherit it. The mean is therefore that common rate, and the test asks whether the effective extracellular rate is zero: the compartment is caseous, the caseation count is at least 14, or the live count is at least 115 (S4.6). The caseation disjunct is redundant in the demotion test, since a caseous compartment already has a zero rate, and load-bearing in the promotion test, where it stops a caseous compartment being relabelled replicating.

A compartment with no live bacteria is not grown (S4.4) and keeps its last designation, so a compartment that was arrested and then cleared stays labelled non-replicating, with an empty live list. Intracellular containers are never re-designated, since the pass visits only compartment containers.

The designation changes no behaviour (S4.3). Which designation each M6 consumer reads is specified in M6.11.

Decay

```
decay():
    for each bacterium of the dead list:
        decrement its decay clock if it is positive
        if its clock <= 0:
            delete it and remove it from the dead list
```

The clock is set to `Mtb_Agents_decayTime` = 5 at death (S4.10) and decremented once per call. A bacterium dying on timestep t is deleted on the fifth call after its death, at the end of timestep t + 4. It persists in the dead list for five timesteps, or 50 minutes. Exposure checks are specified in S2.10.

Container	Called from	Frequency
Every compartment container	The extracellular pass, outside the live-count guard	Once per timestep, unconditionally
An infected macrophage's container	The infected rule	Once per timestep
A chronically infected macrophage's container	The chronically infected rule	Once per timestep
A suppressed infected or chronically infected macrophage's container	The suppressed branch	Once per timestep
An activated macrophage's container	Nowhere (S2.8)	Never

Growth and killing are guarded on a positive live count, so a compartment holding only dead bacteria is otherwise untouched by the pass. The unconditional decay call is what lets its dead list drain. Within the pass, growth and caseum death come first, then the healing test of S5, then decay, so a bacterium killed by the caseum rule on timestep t has its clock decremented for the first time on t.

A dead bacterium is detectable. The exposure rule counts live and dead extracellular bacteria alike (S2.10), and the MHC class II input of M3 counts both over the radius-1 neighbourhood. Dead bacteria are released and dispersed alongside live ones by the macrophage death routes (S4.13). They are excluded from uptake (S4.12), from every growth and killing rule, from the live counts and from the growth-rate and biomass histograms.

S4.15 Disabled module options

M6: antibiotics, resistance and efflux

Absent by default: M6 is off, no antibiotic object exists, and every bacterium's antibiotic concentration list is empty (S4.2). The antibiotic routines of the bacterial code are reached, iterate nothing, make no draw and write nothing. At each division the M6 mutation steps (S4.9) still compute the mutation probability and discard it, and run the mutation loop $\lfloor \mathrm{SF} \rfloor$ times, SF being the scaling factor forwarded by the growth pass (S4.4), floored at 1 by the observation layer; each iteration does nothing. M6 specifies resistance, compensatory mutation, efflux, drug killing and the guard on caseum death (M6.11–M6.14).

M8: randomized growth rate

Absent by default: growth-rate randomization is an option of M8, selected by a command-line flag that is not set. Under it, the counted constructor draws the stored rate uniformly between per-location bounds, stage 1 of the dispatch reads that rate (S4.5), and the write-back compounds the environmental modulation across steps (M8.7).

S4.16 Parameters

No parameter of this submodel is sampled across the virtual library. Every bacterium in every run has the same intrinsic properties.

Identifier	Value	Units	Meaning
`Mtb_Agents_growthRateIntMtb`	0.0045	fraction per timestep	Intrinsic intracellular rate (S4.2, S4.5)
`Mtb_Agents_growthRateExtMtb`	0.0025	fraction per timestep	Intrinsic extracellular rate (S4.2, S4.5)
`Mtb_Agents_growthRateIntMtbMin`	0.001	fraction per timestep	Lower randomization bound, intracellular (M8; inert by default); lower bound of the intracellular growth-rate histogram (S9)
`Mtb_Agents_growthRateIntMtbMax`	0.005	fraction per timestep	Upper bound, as for the minimum
`Mtb_Agents_growthRateExtMtbMin`	0.0005	fraction per timestep	Lower randomization bound, extracellular (M8; inert by default); lower bound of the extracellular growth-rate histogram (S9)
`Mtb_Agents_growthRateExtMtbMax`	0.00325	fraction per timestep	Upper bound, as for the minimum
`Mtb_Agents_initialBiomass`	1	biomass	Biomass of a constructed bacterium (S4.2)
`Mtb_Agents_divisionThreshold`	2	biomass	Division when biomass $\geq$ this value (S4.8); upper bound of the biomass histogram (S9)
`Mtb_Agents_deathThreshold`	0.5	biomass	Starvation when biomass < this value; never satisfied (S4.8); lower bound of the biomass histogram (S9)
`Mtb_Agents_divisionFraction`	0.43	fraction	Share of the pre-division biomass taken by one product (S4.9)
`Mtb_growthExtMtbBound`	114.527	bacteria	Crowding denominator (S4.6); caseum-death threshold (S4.11); exceedance statistic (S9)
`Mtb_Agents_growthRateExtMtbCaseatedFrac`	0.23	fraction	Fraction for the reduction threshold[1]
`Mtb_Agents_noGrowthExtMtbCaseatedFrac`	0.56	fraction	Fraction for the arrest threshold[1]

Identifier	Value	Units	Meaning
`Mtb_Agents_noGrowthExtMtbCaseatedRadius`	2	compartments	Radius used in the threshold computation only[2]
`Mtb_Agents_growthRateExtMtbCaseated`	5, computed	count	Reduction threshold on c (S4.6)[1]
`Mtb_Agents_noGrowthExtMtbCaseated`	14, computed	count	Arrest threshold on c (S4.6)[1]
`Mtb_Agents_decayTime`	5	timesteps (50 min)	Decay clock set at death (S4.10, S4.14)
`Mtb_Agents_growthRateFactorPostAdaptiveIntMtb`	1.0	dimensionless	Intracellular rate multiplier after onset (S4.7)
`Mtb_Agents_growthRateFactorDelayIntMtb`	0.0	timesteps	Delay of that onset after `Recruit_Prob_timeRecEnabled` (S4.7)

[1] The two thresholds are not taken from the parameter file but derived at load time as $\theta = \lfloor \phi(2r+1)^D \rfloor$, with r = 2 and D = 2, a neighbourhood of 25 compartments: $\lfloor 0.23 \times 25 \rfloor = \lfloor 5.75 \rfloor = 5$ and $\lfloor 0.56 \times 25 \rfloor = 14$. The parameter file supplies 0 for both, and both are overwritten. Taken literally, a threshold of 0 would put every compartment in the arrest branch of S4.6 and stop all extracellular growth, so a reimplementation must perform the derivation or supply the integers. Startup checks require that each no-growth quantity exceed its growth-reduction counterpart, checked separately on the fractions and on the computed integers, with the integer check exempted when either value is 0. The fraction check guards against an inverted specification, and the integer check against two fractions truncating to one integer. Both are satisfied, since 0.23 < 0.56 and 5 < 14. The exponent is the dimension count, so the thresholds change under M5.

[2] The radius parameter is read only by the threshold computation. The radius of the sweep producing the count those thresholds are compared with is fixed in the grid at 2 (S5.5), so changing the parameter would alter the thresholds without altering the neighbourhood.

The division threshold is exactly twice the initial biomass, so a founder bacterium divides after one biomass doubling.

Fixed constants: the sweep radius, 2; the divisor in the caseum death exponent, 10; the asymmetry coin, 0.5. The probability 0.1 of killing at a caseous neighbour is fixed in S2.8, and the caseum secretion factor 0.25 in S5.6. There is no per-location decay time: dead bacteria persist for five timesteps whether intracellular or extracellular.

Read but without effect by default: `Mtb_Agents_growthRateFactorPostAdaptiveIntMtb` and `Mtb_Agents_growthRateFactorDelayIntMtb` (an identity and its onset); `Mtb_Agents_deathThreshold` (read on every growth step, never satisfied, and not dead code: a change to the division fraction, the threshold or any mechanism allowing a negative rate would make it operative); the four randomization bounds in their randomization role, though they are active as histogram bounds; `Mtb_Agents_initialReserve` = 0 (sets a field that is never updated); `Mtb_Agents_maxReserve` = 1 (read only by the output writer, which never emits reserve columns, since the reserve sample count is zeroed); `Mtb_Agents_degRateBiomass` = 0.001 and `Mtb_Agents_degRateBiomassMin` = 0 (read by nothing); `Mtb_minGrowthBin` = 0.0 and `Mtb_maxGrowthBin` = 0.006 (read by nothing; the growth-rate histogram bounds are the four randomization parameters); `NFkB_meanNFkB` = 10^5 (governs the log-normal construction draw, whose result is never read; the draw itself belongs to the random stream).

The parameters of the disabled modules M6 and M8 are indexed in Appendix B.

S4.17 Interfaces

With	Provides or receives	See
Spine 2.5	Inventory of bacterial state variables	Spine 2.5
Spine Element 3	Position of the extracellular pass in the cellular timestep	Spine Element 3
Spine 5.4	The seeding macrophage's bacterium; parameter-file extracellular bacteria	Spine 5.4
S2	Receives uptake, killing, intracellular growth and decay calls from the macrophage rules; S2 holds the fate of bacteria by death route, the activated macrophage's lack of a decay call and the caseous-neighbour kill probability	S2.5–S2.10
S3	T cells kill macrophages, never bacteria; the bacterial outcome follows the macrophage death route	S3, S4.13
S5	Provides the caseation status and the caseation count; the sweep also performs entrapment; the healing test runs inside the extracellular pass; S5 specifies the hard-coded sweep radius and the sweep-order dependence of the count	S5.3–S5.5
S6	None: bacteria secrete nothing and apply no sink; the caseum secretion factor is in S5.6	S6
S7	None: bacteria do not read oxygen	S7
S9	Bacterial statistics, histograms and the outside-granuloma release count	S9
M1	None: bacteria read and write no field, so neither the solver nor its cadences affect them	M1
M2	The intracellular modulator reads `Recruit_Prob_timeRecEnabled` whichever recruitment option is selected; under the lymph-node option the switch fires at that parameter's time, independently of T cell arrival; inert by default	M2
M3	Live and dead bacteria feed the MHC class II input of the disabled pathway	M3
M4	None: TGF-β acts on cells only	M4
M5	The radius-2 neighbourhood, the computed thresholds and the dispersal neighbourhood change with the dimension; the crowding bound does not	M5
M6	Resistance, compensatory mutation, efflux, drug killing and the guard on caseum death; off by default (S4.15)	M6
M7	None: anti-TNF agents act on the TNF field and on cells	M7

With	Provides or receives	See
M8	Growth-rate randomization replaces the stage 1 base rate (S4.15); reinfection places a second infected macrophage whose one bacterium is built by the counted constructor, with the intracellular triple and the four construction draws, founding an independent lineage; disabled by `Core_reinfectionTimestep` = -1; the other M8 options have no bacterial effect	M8
M9	Neutrophil-held containers grow and divide by S4.4 to S4.9 and decay by S4.14	M9.6
Appendix B	Toggles and inert parameters	Appendix B

Observation

The bacterial statistics are accumulated by the containers, not by the bacteria, during macrophage state commitment and creation and in the per-compartment sweep of the statistics block. S9 owns them: the standing counts, the non-replicating count, the mortality categories, the exceedance count, the division counter and the per-bacterium histograms. S4 provides the quantities they read: live counts, biomass, stored rates, the mean growth rate (S4.3), the starvation and division totals of the growth pass (S4.4), and the return values of the death and dispersal routines (S4.10, S4.11, S4.13). No statistic counts the standing dead population.