Lymph node swelling combined with temporary effector T cell retention aids T cell response in a model of adaptive immunity

Supplementary File 2: Supplementary Methods B

This section presents the logic underlying the sub-functions called during the simulation. The order each method is called is depicted in the 'Main' diagram (Fig 2B) and summarised below. Properties stored in T cells and DCs are shown in S1 File, Fig C.

- Each time-step, in the context class, the number of T cells to add and the value of properties are calculated. Positions to place T cells are selected from a list of possible coordinates and new instances (copies) of the relevant T cell class. (eg. Cognate or Non-cognate) are produced.
- Calling the **'UpDateTCs'** function applies updates to the history of all agent's present, eg. how long since they have entered.
- Subfunctions that result in removal of an agent, for example, a T cell dying (**UpdateTCsAgeing, RemoveTCsAgeing**), also requires removal of any links **RemoveLinks** (projections) to DCs, and the update of relevant DC properties, eg. number of T cells bound.
- T cells move (**TCsMove**), and any DCs to enter are added (**AddDCs**).
- **UpdateDCs** includes updating MHC expression, DC states, such as whether or not they are licenced, and timecounts, such as duration of links to T cells. Interactions are ended by removing Links (**removeLinks**) after checking rules that take into account T cell type and interaction history. The DCs are aged (AgeDCs) and may be removed, along with any corresponding links.
- Cognate T cells then undergo updates (**UpdateCognateTCs**) including updating age and history counts, gaining or losing stimulation, and assessing probability of, then carrying out activation/differentiation and proliferation.
- Finally, functions are called in the Context class that update counters of inflammation state (sum of MHC values), count T cells present, and adjust the volume and geometry of the LN accordingly.

2.1 UML diagrams describing the rules used to build the ABM

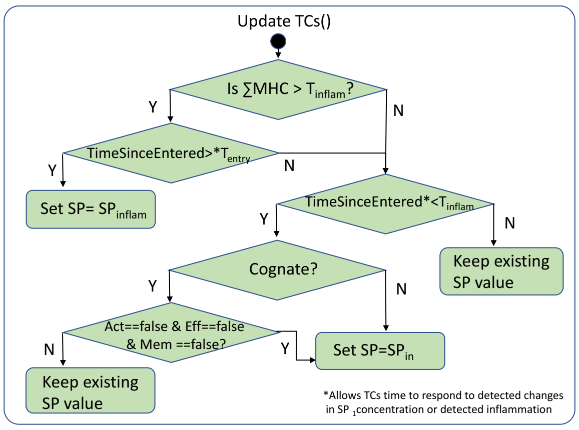
Update TCs()
Is ∑MHC > T_inflam?
Y
N
TimeSinceEntered>*T_entry
Y
N
Set SP= SP_inflam
TimeSinceEntered*<T_inflam
Y
N
Keep existing SP value
Cognate?
Y
N
Act==false & Eff==false & Mem ==false?
N
Y
Set SP=SP_in
Keep existing SP value
*Allows TCs time to respond to detected changes in SP_1 concentration or detected inflammation

Figure 1: **UML diagram for 'UpdateTCs'.** By checking the time since the TC entered, a 'lag time' representing time for detection of signals and receptor internalisation is permitted. If the cell is newly entered, and not an activated or effector TC, $S1P_1r$ is down-regulated to the value of SP_{in}

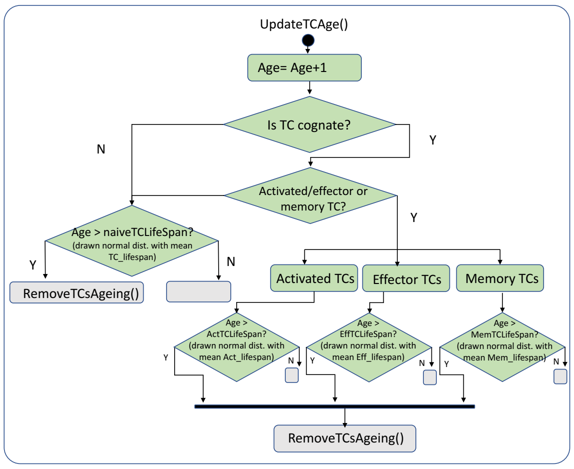
UpdateTCAge()
Age= Age+1
Is TC cognate?
N
Y
Activated/effector or memory TC?
Y
Age > naiveTCLifeSpan? (drawn normal dist. with mean TC_lifespan)
Y
N
RemoveTCsAgeing()
Activated TCs
Effector TCs
Memory TCs
Age > ActTCLifeSpan? (drawn normal dist. with mean Act_lifespan)
Age > EffTCLifeSpan? (drawn normal dist. with mean Eff_lifespan)
Age > MemTCLifeSpan? (drawn normal dist. with mean Mem_lifespan)
Y
N
RemoveTCsAgeing()

Figure 2: **UML diagram for 'UpdateTCsAgeing'.** Lifespan is dependent on TC state (see S1 File, Table B).

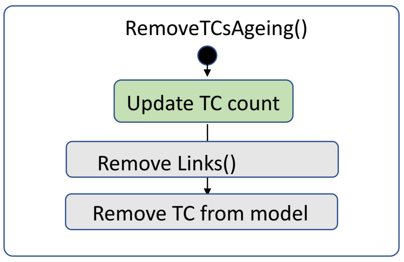

Figure 3: UML diagram for 'RemoveTCsAgeing'. The context stores a counter of how many TCs are present which is updated when a T cell is removed. Any links to DCs are also removed.

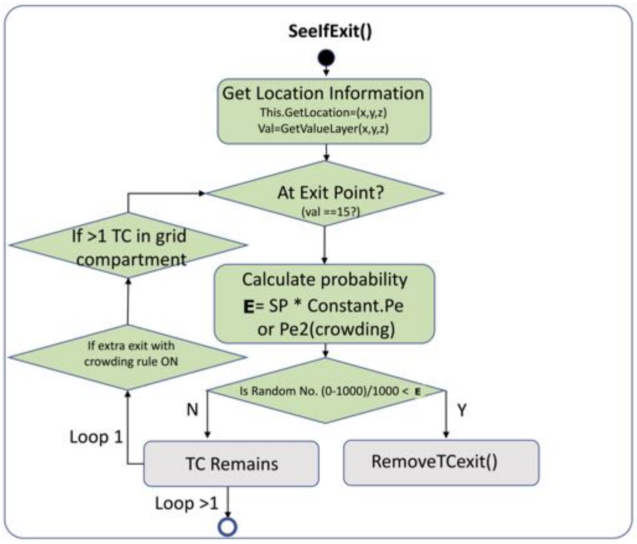

Figure 4: UML diagram for 'SeeIfExit'. T cells check their environment by checking the 'valueLayer'. If in an exit area, the probability of exit (E) is calculated by multiplying a constant Pe by the individual TCs relative S1P1r expression. A random number between 0 and 1 is generated and if less than E, the T cell will exit. A second check of TC crowding (crowding rule) is applied in some simulations.

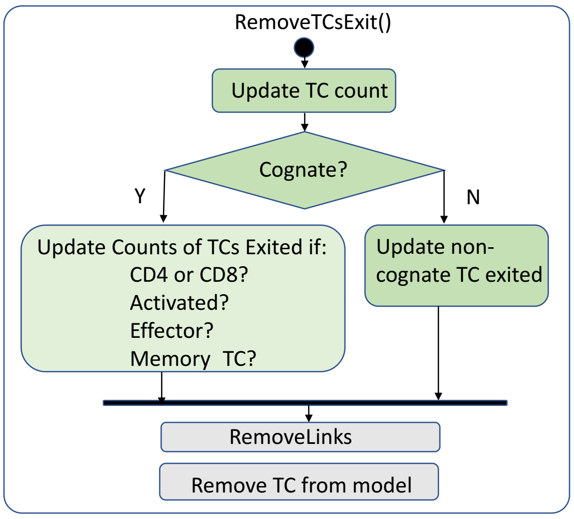

Figure 5: UML diagram for 'RemoveTCExit'. Ageing and TC exit are treated differently due to different counters (stored as Context properties) being updated.

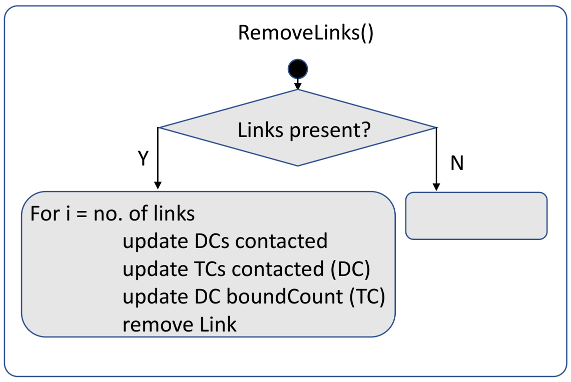

Figure 6: UML diagram for 'RemoveLinks'. Links between TCs and DCs are formed using a projection network, a feature of RepastSimphony. Before removal, properties within the linked agents relating to interactions are updated.

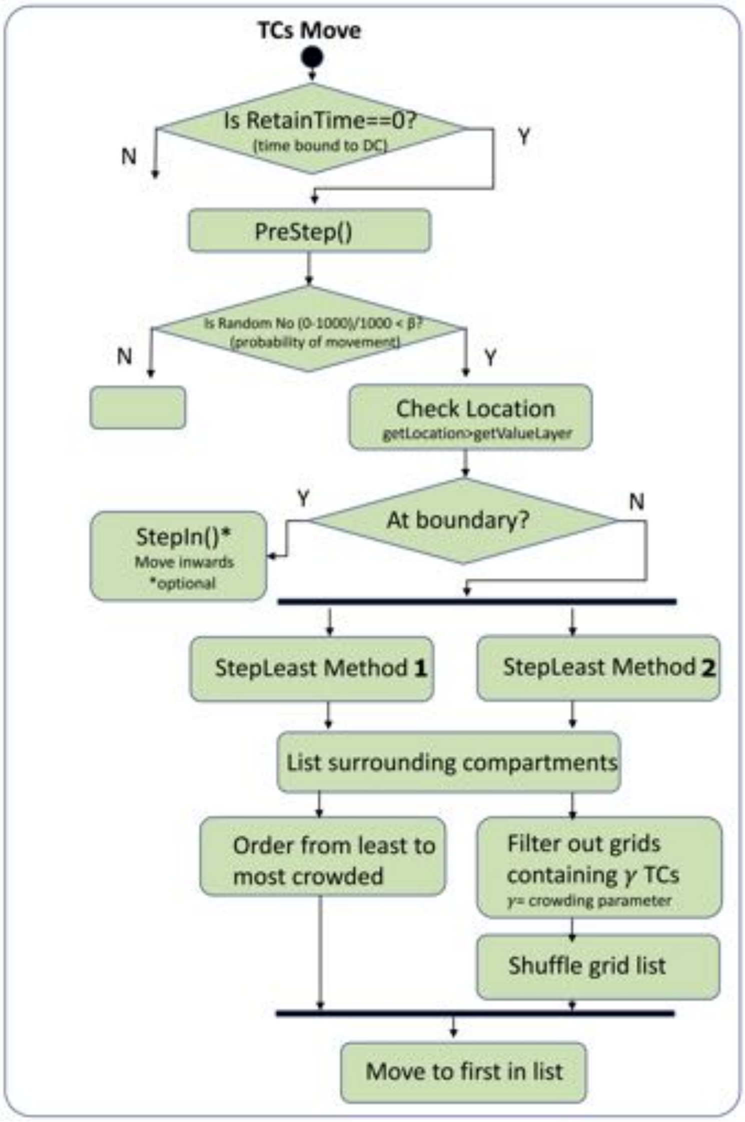

Figure 7: UML diagram for 'TCsMove'. Only unbound T cells will migrate. After determining whether the cell will move this time-step (using probability of movement β), the model has the capacity for different movement options. The GridCell class makes it possible to provide a list of neighbouring compartments and their contents. Method 1 involves iterating over the list and moving to the least crowded neighbouring compartment. Method 2 filters the list, considering the maximum TCs permitted per grid compartment (dictated by crowding parameter γ), then randomely selects from the remaining positions.

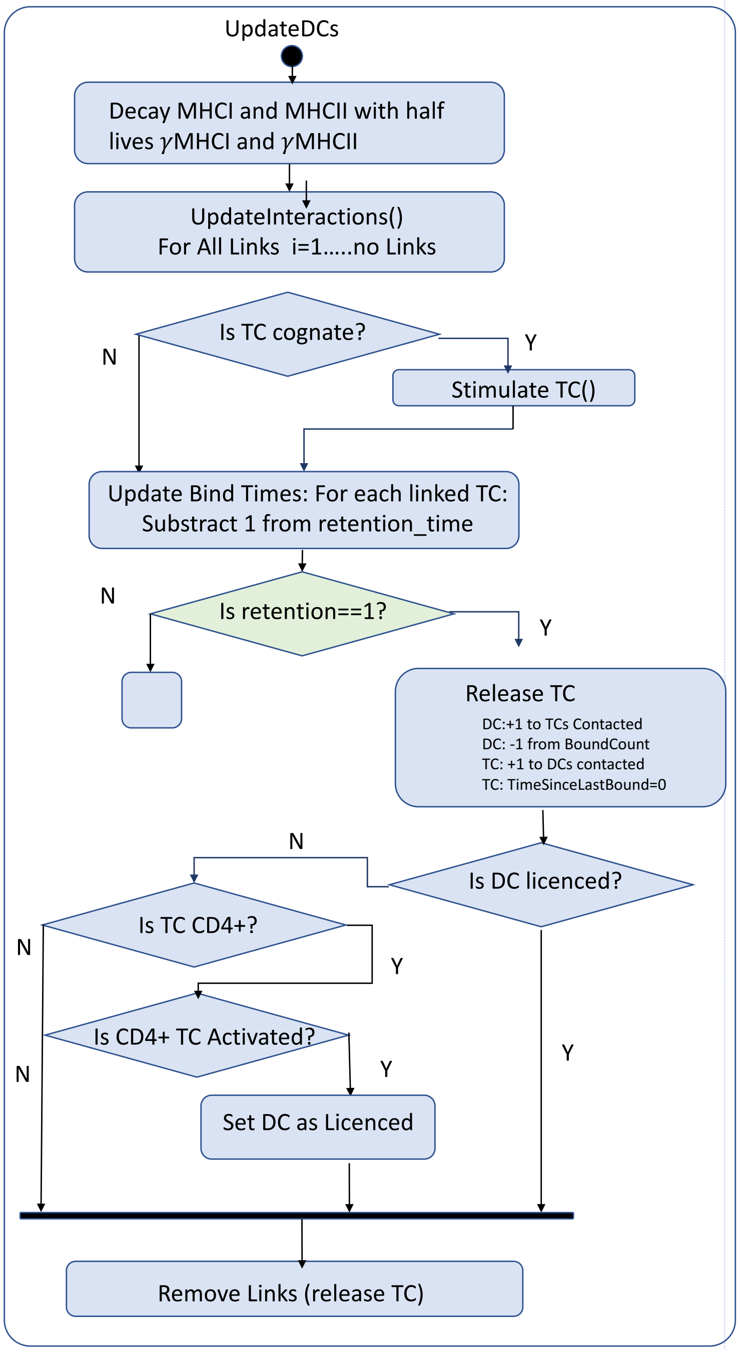

Figure 8: UML diagram for 'UpdateDCs'. DCs express a decaying MHC signal, accessible by TCs that they have formed links with. Each interaction (link) is designated a duration to exist (retention), which is reduced every timestep. When retention falls to 1, the link is removed. DCs enter as mature migrating DCs but may become licenced through interaction with activated CD4+ TCs. This means the DC is more effective at activating CD8+ T cells (implemented by switching activation probability curve parameters to equal those used when interacting with CD4+ TCs).

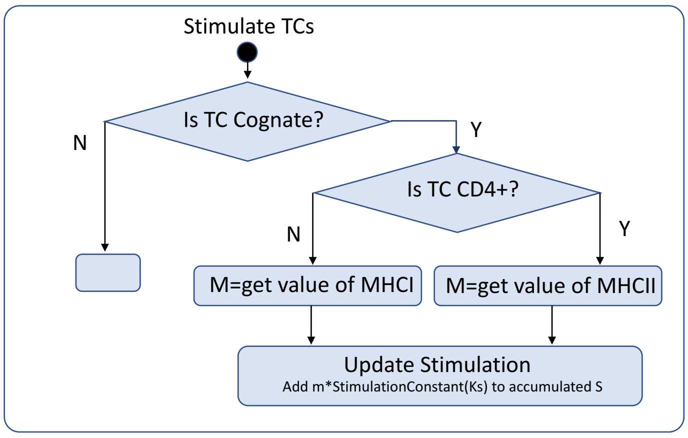

Figure 9: UML diagram for ’StimulateTCs’. DCs can access agent properties that they link with (interact with) and alter the agent’s accumulated stimulation based on the DCs MHCI/II presentation and Stimulation constant, Ks.

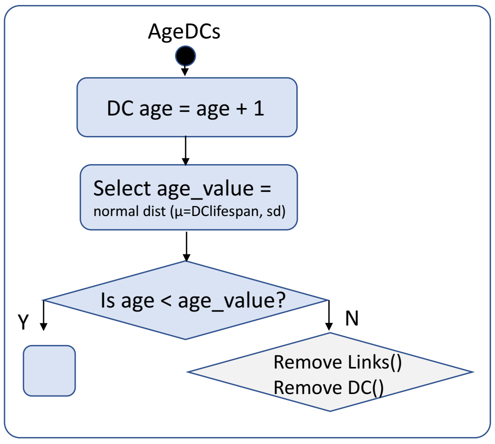

Figure 10: UML diagram for ’AgeDCs’. Mature DCs have a lifespan of around 60hours (+/-2.7hrs), drawn from a normal distribution. When reached, any interacting agents are notified, any links are removed, and the DC is removed.

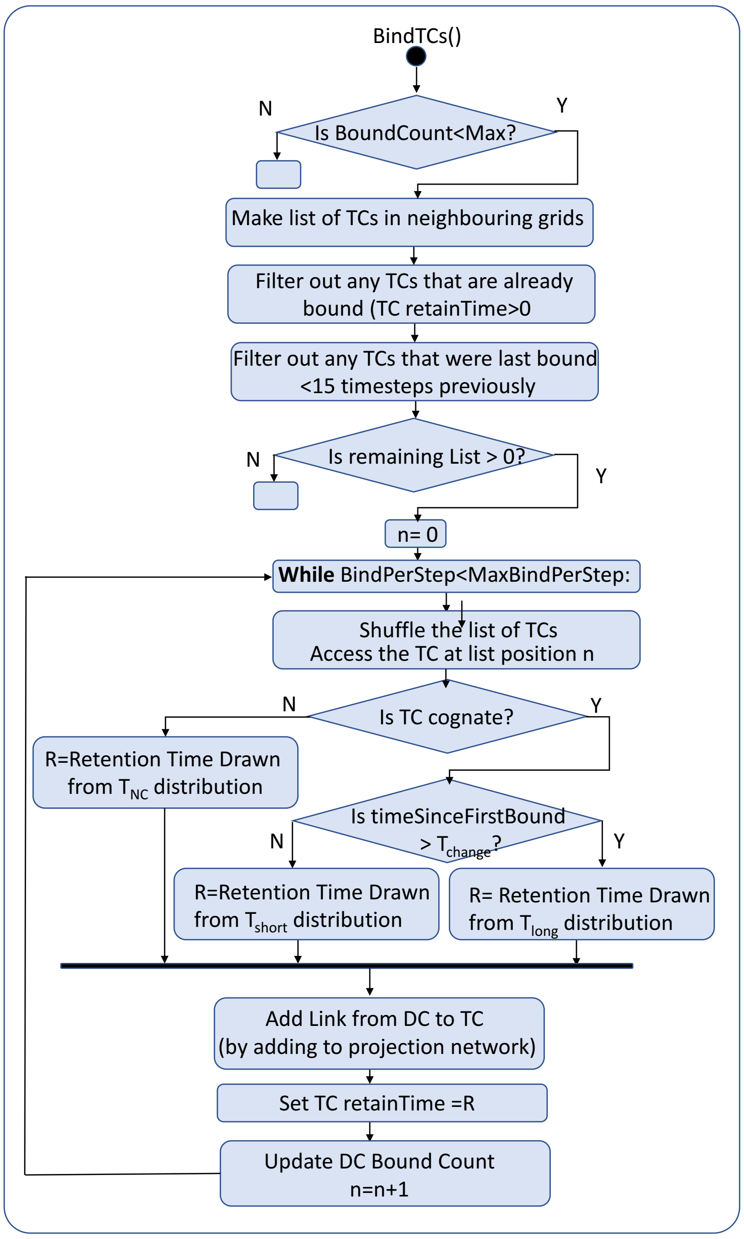
BindTCs()
Is BoundCount<Max?
N
Y
Make list of TCs in neighbouring grids
Filter out any TCs that are already bound (TC retainTime>0
Filter out any TCs that were last bound <15 timesteps previously
Is remaining List > 0?
N
Y
n= 0
While BindPerStep<MaxBindPerStep:
Shuffle the list of TCs
Access the TC at list position n
Is TC cognate?
N
Y
R=Retention Time Drawn from T_{NC} distribution
Is timeSinceFirstBound > T_{change}?
N
Y
R=Retention Time Drawn from T_{short} distribution
R= Retention Time Drawn from T_{long} distribution
Add Link from DC to TC (by adding to projection network)
Set TC retainTime =R
Update DC Bound Count
n=n+1

Figure 11: UML diagram for 'BindTCs'. When setting up new interactions, the DCs must first not exceed their maximum number of interactions (BoundCount). The GridCell property that can provide a list of neighbouring compartments and their agents is used again to determine TCs eligable to interact. Once selected, the TCs receive a interaction time stored as a 'retention' property drawn from a relevant probability density function, based on their type and history. This property decreases each timestep, until the value reaches 1 and the T cell is released.

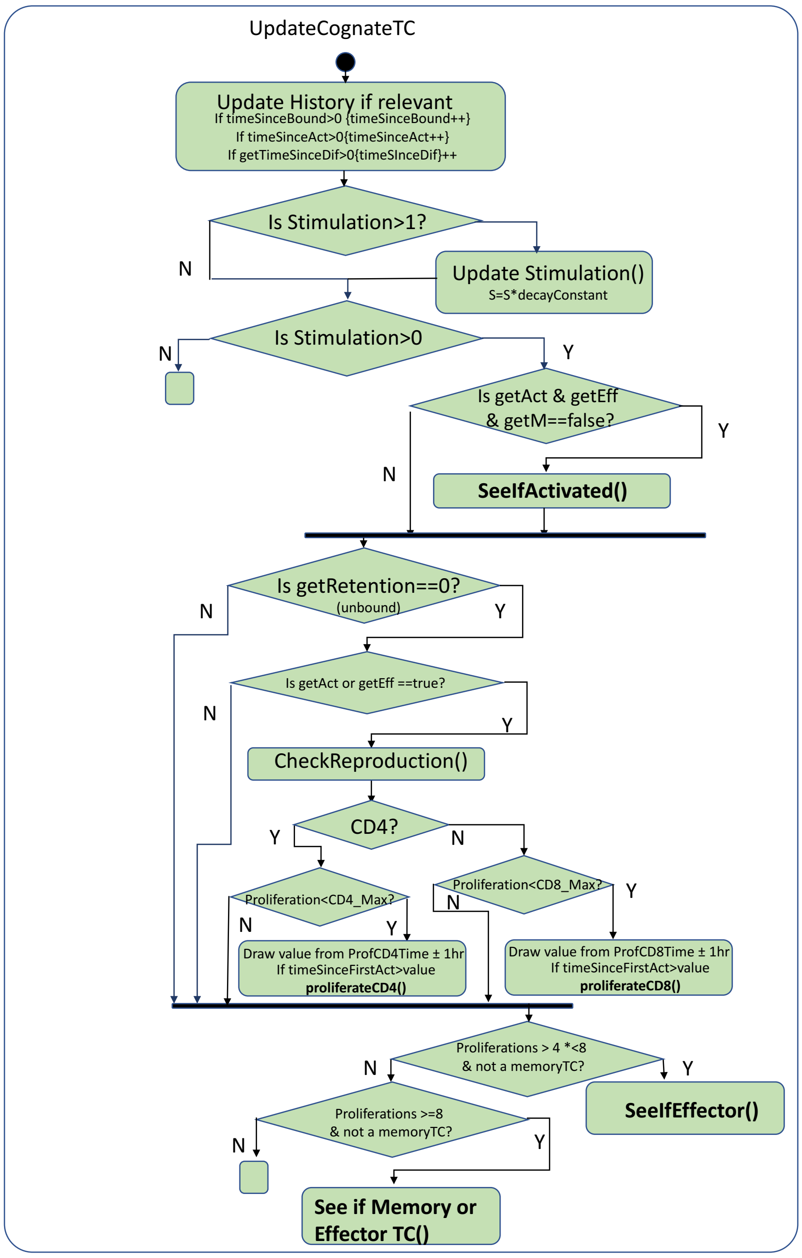
UpdateCognateTC
Update History if relevant
If timeSinceBound>0 {timeSinceBound++}
If timeSinceAct>0{timeSinceAct++}
If getTimeSinceDif>0{timeSInceDif}++
Is Stimulation>1?
N
Update Stimulation()
S=S*decayConstant
Is Stimulation>0
N
Y
Is getAct & getEff & getM==false?
Y
N
SeeIfActivated()
Is getRetention==0? (unbound)
N
Y
Is getAct or getEff ==true?
N
Y
CheckReproduction()
CD4?
Y
N
Proliferation<CD4_Max?
N
Y
Proliferation<CD8_Max?
N
Y
Draw value from ProfCD4Time ± 1hr If timeSinceFirstAct>value proliferateCD4()
Draw value from ProfCD8Time ± 1hr If timeSinceFirstAct>value proliferateCD8()
Proliferations > 4 *<8 & not a memoryTC?
N
Y
SeeIfEffector()
Proliferations >=8 & not a memoryTC?
N
Y
See if Memory or Effector TC()

Figure 12: UML diagram for 'UpdateCognateTCs'.
Cognate TCs store properties determining their current state and history. Stimulation is increased during interaction with DCs, and decays each time step. Activation probability is a sigmoidal function of Stimulation (Figure 14). Proliferation and differentiation are permitted only when T cells are unbound. CD4+ and CD8+ T cells follow different rules, using sigmoidal functions that are parameterized differently. SeeIfEffector or Memory use accumulated stimulation as inputs to sigmoidal functions to determine if the cell will differentiate , then a probability ratio to determine which cell type will be produced.

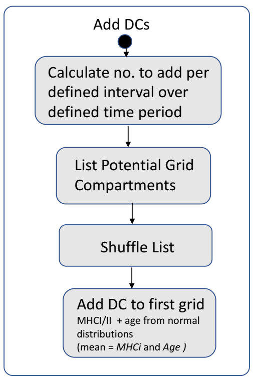
Add DCs
Calculate no. to add per defined interval over defined time period
List Potential Grid Compartments
Shuffle List
Add DC to first grid
MHCI/II + age from normal distributions
(mean = MHCi and Age)

Figure 13: UML diagram for 'AddDCs'.

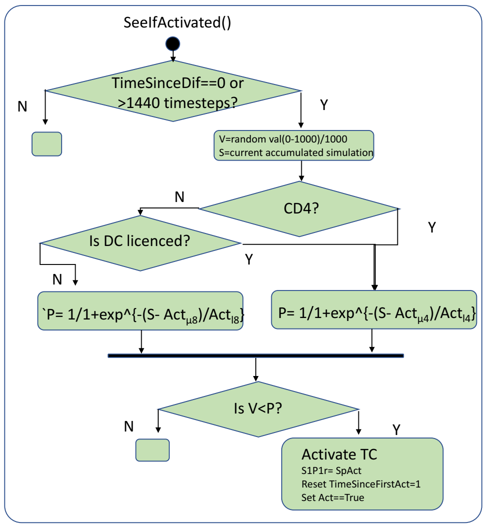
SeeIfActivated()
TimeSinceDif==0 or >1440 timesteps?
N
Y
V=random val(0-1000)/1000
S=current accumulated simulation
CD4?
N
Y
Is DC licenced?
N
Y
`P= 1/1+exp^{-(S- Actμ8)/Actl8}
P= 1/1+exp^{-(S- Actμ4)/Actl4}
Is V<P?
N
Y
Activate TC
S1P1r= SpAct
Reset TimeSinceFirstAct=1
Set Act==True

Figure 14: UML diagram for 'SeeIfActivated'. Stimulation is used as an input to sigmoidal functions that produce a probability of activation. If generation of a random decimal number between 0 and 1 is less than the probability of activation, then the T cell is activated and S1P1r expression is down-regulated.

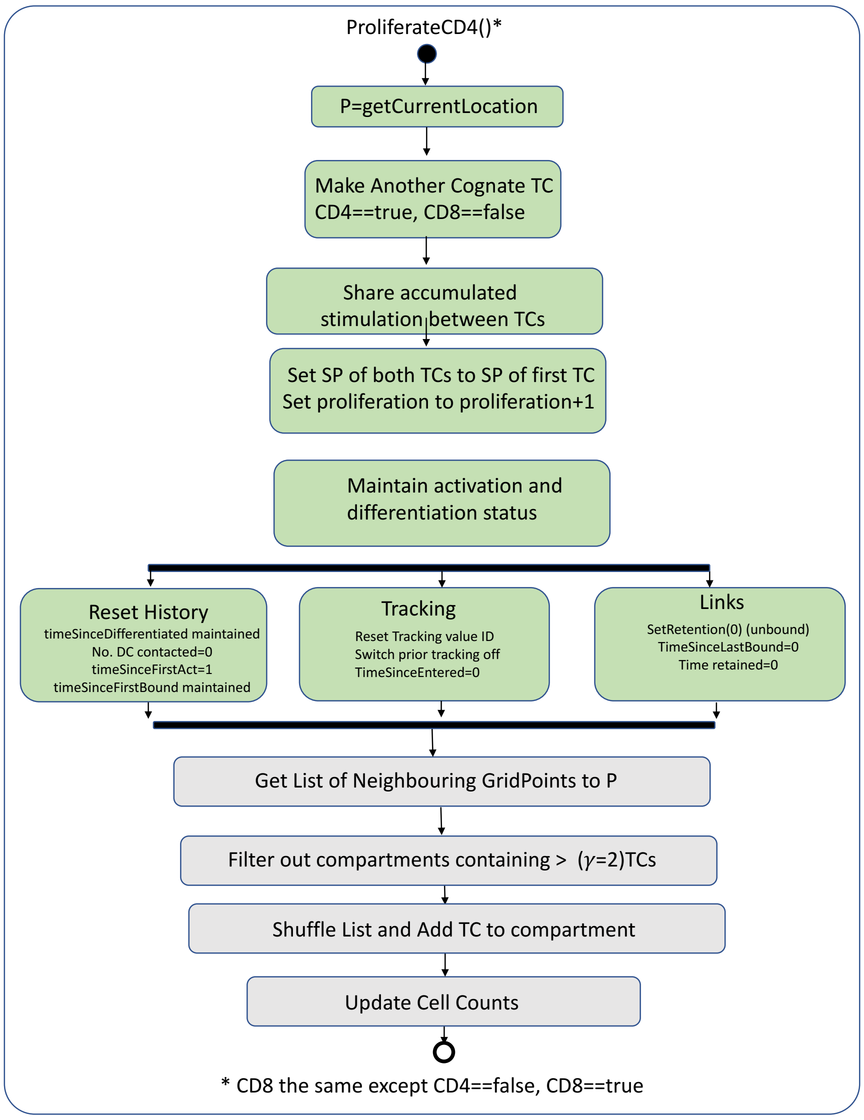
ProliferateCD4()*
P=getCurrentLocation
Make Another Cognate TC CD4==true, CD8==false
Share accumulated stimulation between TCs
Set SP of both TCs to SP of first TC Set proliferation to proliferation+1
Maintain activation and differentiation status
Reset History
timeSinceDifferentiated maintained
No. DC contacted=0
timeSinceFirstAct=1
timeSinceFirstBound maintained
Tracking
Reset Tracking value ID
Switch prior tracking off
TimeSinceEntered=0
Links
SetRetention(0) (unbound)
TimeSinceLastBound=0
Time retained=0
Get List of Neighbouring GridPoints to P
Filter out compartments containing > (γ=2)TCs
Shuffle List and Add TC to compartment
Update Cell Counts
* CD8 the same except CD4==false, CD8==true

Figure 15: UML diagram for 'Proliferation'. Proliferation of T cells is complex because accumulated Stimulation is shared between the daughter cells (which is in-fact a reset existing cell and a copy). History, tracking value and links are updated to ensure that the two new cells have individual properties

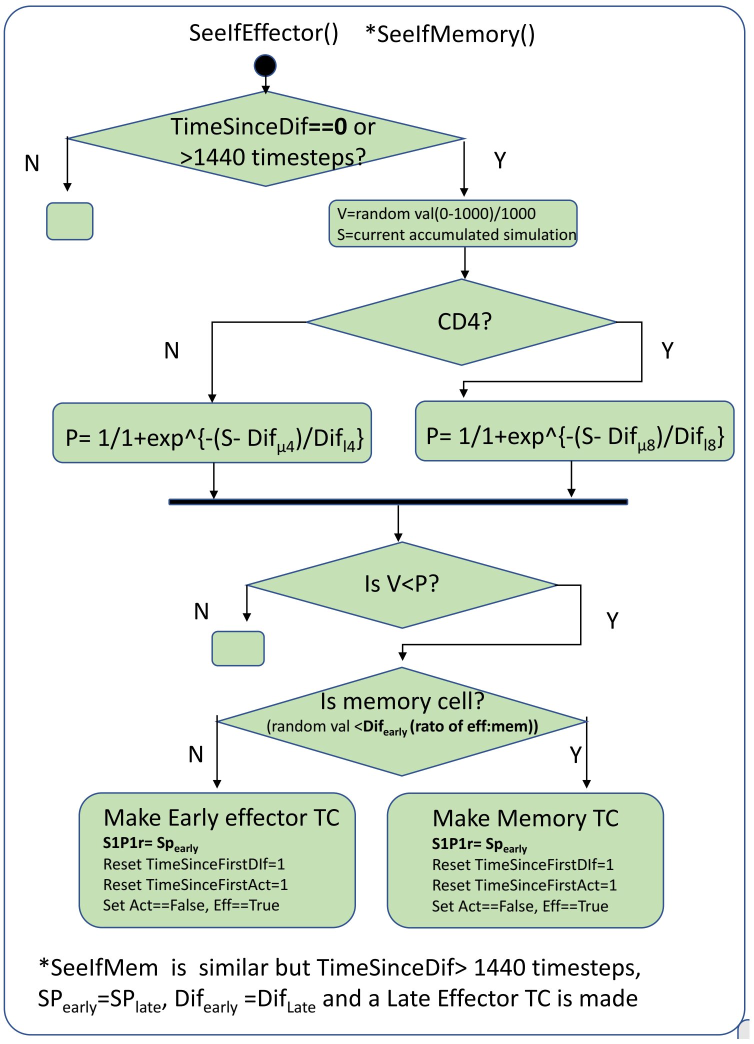
SeeIfEffector() *SeeIfMemory()
TimeSinceDif==0 or >1440 timesteps?
N
Y
V=random val(0-1000)/1000
S=current accumulated simulation
CD4?
N
Y
P= 1/1+exp^{-(S- Dif$_{\mu4}$)/Dif$_{l4}$}
P= 1/1+exp^{-(S- Dif$_{\mu8}$)/Dif$_{l8}$}
Is V<P?
N
Y
Is memory cell?
(random val <Dif$_{early}$ (rato of eff:mem))
N
Y
Make Early effector TC
S1P1r= Sp$_{early}$
Reset TimeSinceFirstDIf=1
Reset TimeSinceFirstAct=1
Set Act==False, Eff==True
Make Memory TC
S1P1r= Sp$_{early}$
Reset TimeSinceFirstDIf=1
Reset TimeSinceFirstAct=1
Set Act==False, Eff==True
*SeeIfMem is similar but TimeSinceDif> 1440 timesteps,
SP$_{early}$=SP$_{late}$, Dif$_{early}$ =Dif$_{Late}$ and a Late Effector TC is made

Figure 16: UML diagram for 'SeeIfEffector' or 'SeeIfEffectorMemory'. The two behaviours are similar but for seeIfEffectorMemory, the time since differentation cannot be zero, as the TC must have previously differentiated, and more memory TCs are made.

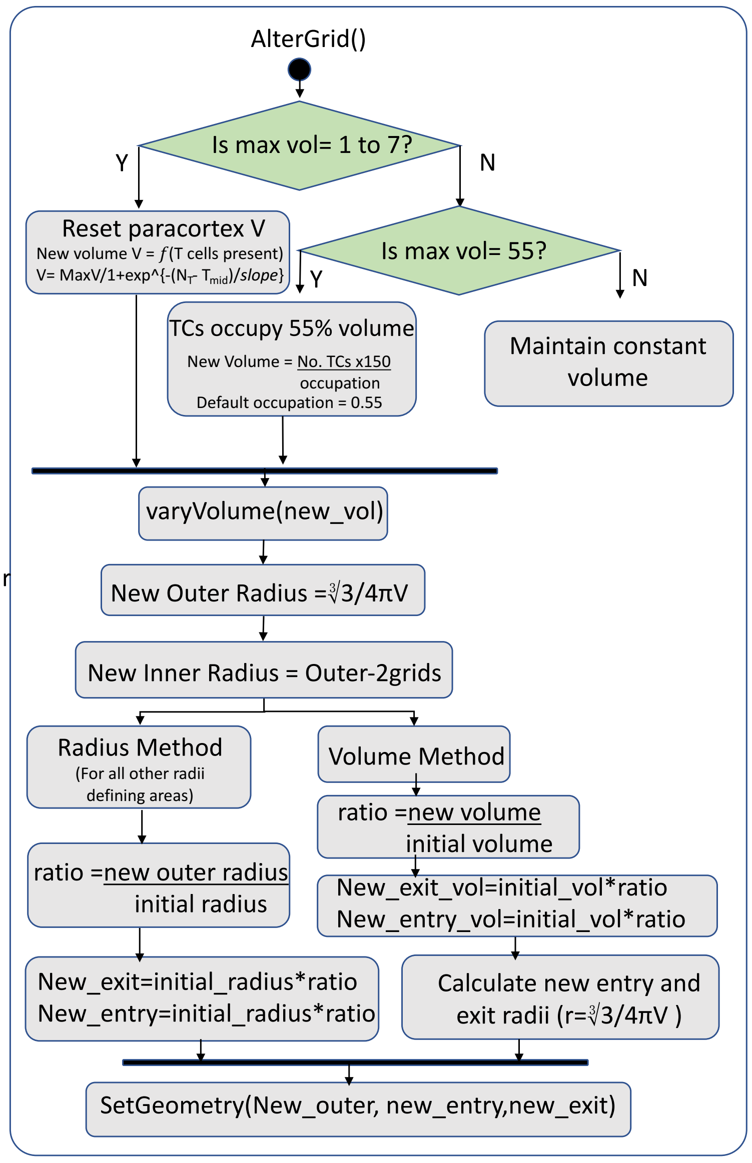
AlterGrid()
Is max vol= 1 to 7?
Y
N
Reset paracortex V
New volume V = ƒ(T cells present)
V= MaxV/1+exp^{-(N_T- T_mid)/slope}
Is max vol= 55?
Y
N
TCs occupy 55% volume
New Volume = No. TCs x150 / occupation
Default occupation = 0.55
Maintain constant volume
varyVolume(new_vol)
New Outer Radius =∛3/4πV
New Inner Radius = Outer-2grids
Radius Method
(For all other radii defining areas)
Volume Method
ratio =new volume / initial volume
ratio =new outer radius / initial radius
New_exit_vol=initial_vol*ratio
New_entry_vol=initial_vol*ratio
New_exit=initial_radius*ratio
New_entry=initial_radius*ratio
Calculate new entry and exit radii (r=∛3/4πV)
SetGeometry(New_outer, new_entry,new_exit)

Figure 17: UML diagram for 'AlterGrid'. This methods carries out the calculation of the new paracortex volume then passes the calculated radii to 'Set Geometry'.

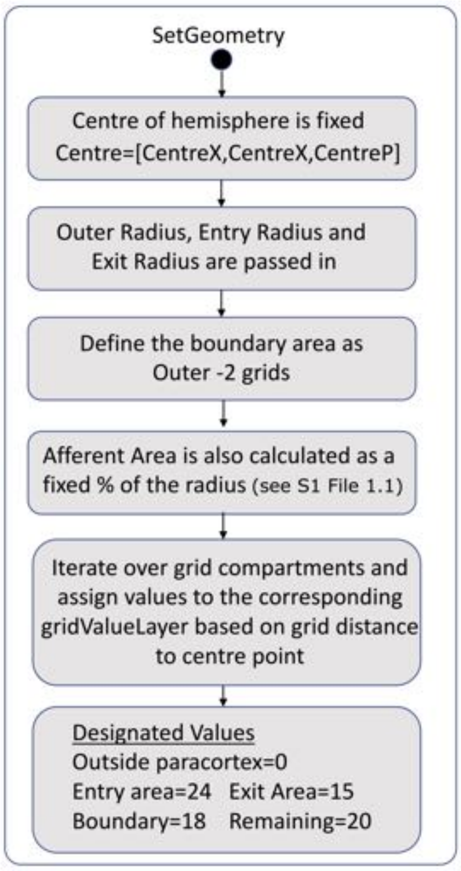
SetGeometry
Centre of hemisphere is fixed
Centre=[CentreX,CentreX,CentreP]
Outer Radius, Entry Radius and Exit Radius are passed in
Define the boundary area as Outer -2 grids
Afferent Area is also calculated as a fixed % of the radius (see S1 File 1.1)
Iterate over grid compartments and assign values to the corresponding gridValueLayer based on grid distance to centre point
Designated Values
Outside paracortex=0
Entry area=24 Exit Area=15
Boundary=18 Remaining=20

Figure 18: UML diagram for 'SetGeometry'. Only the radii are required as inputs to determine the geometry of the LN as described in S1 File. 1.1