DNA Damage & DNA Repair module

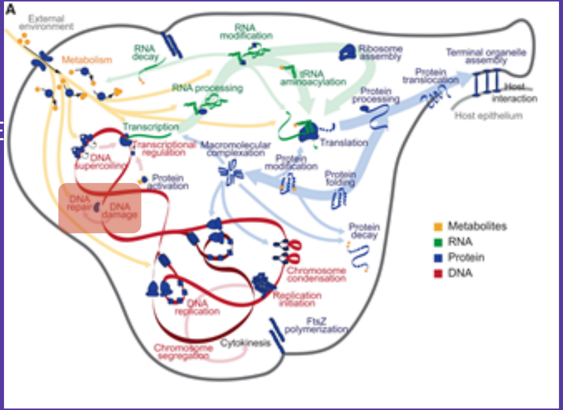
Arne Bittig – University of Rostock

Audald Lloret-Villas – EMBL-EBI

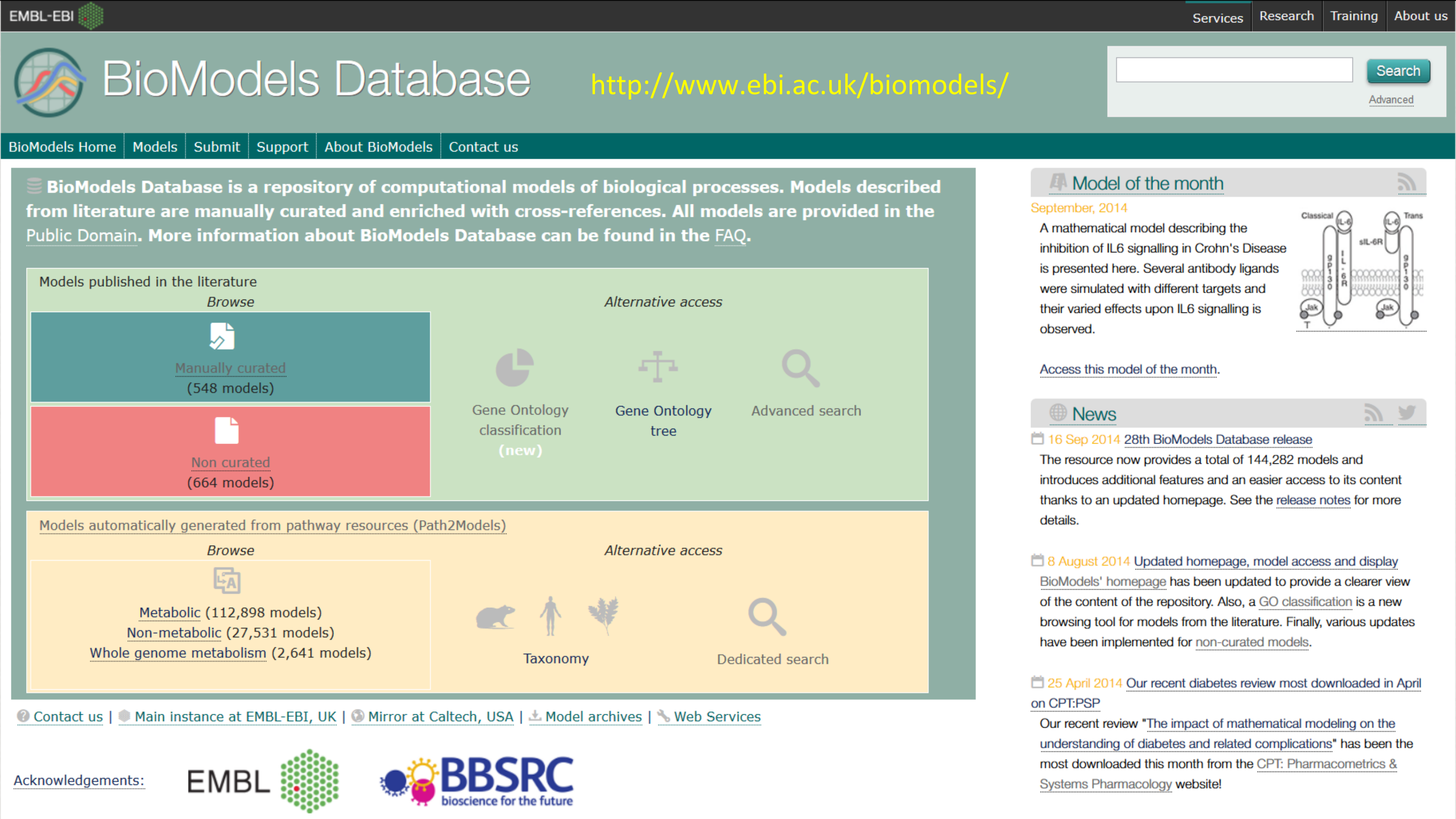
Mahesh Sharma –National Institute of Pharmaceutical Education and Research (NIPE

Namrata Tomar – University of Erlangen

Viji Chelliah (EMBL-EBI)



Whole-cell summer school, 9th-13th March – Rostock | Germany



EMBL-EBI

Services | Research | Training | About us

BioModels Database

http://www.ebi.ac.uk/biomodels/

Search

Advanced

BioModels Home | Models | Submit | Support | About BioModels | Contact us

BioModels Database is a repository of computational models of biological processes. Models described from literature are manually curated and enriched with cross-references. All models are provided in the Public Domain. More information about BioModels Database can be found in the FAQ.

Models published in the literature

Browse

Manually curated (548 models)

Non curated (664 models)

Alternative access

Gene Ontology classification (new)

Gene Ontology tree

Advanced search

Models automatically generated from pathway resources (Path2Models)

Browse

Metabolic (112,898 models)

Non-metabolic (27,531 models)

Whole genome metabolism (2,641 models)

Alternative access

Taxonomy

Dedicated search

Contact us | Main instance at EMBL-EBI, UK | Mirror at Caltech, USA | Model archives | Web Services

Acknowledgements: EMBL BBSRC bioscience for the future

Model of the month

September, 2014

A mathematical model describing the inhibition of IL6 signalling in Crohn's Disease is presented here. Several antibody ligands were simulated with different targets and their varied effects upon IL6 signalling is observed.

Access this model of the month.

News

16 Sep 2014 28th BioModels Database release

The resource now provides a total of 144,282 models and introduces additional features and an easier access to its content thanks to an updated homepage. See the release notes for more details.

8 August 2014 Updated homepage, model access and display

BioModels' homepage has been updated to provide a clearer view of the content of the repository. Also, a GO classification is a new browsing tool for models from the literature. Finally, various updates have been implemented for non-curated models.

25 April 2014 Our recent diabetes review most downloaded in April on CPT:PSP

Our recent review "The impact of mathematical modeling on the understanding of diabetes and related complications" has been the most downloaded this month from the CPT: Pharmacometrics & Systems Pharmacology website!

Initial Plan before the start of the summer school

- Identify interface entities (input/output) of the two modules
- Extraction of reactions and building SBGN (using Cell designer/Vanted) map.
- List ChEBI, EC and GO term ids that is needed for annotating the model.

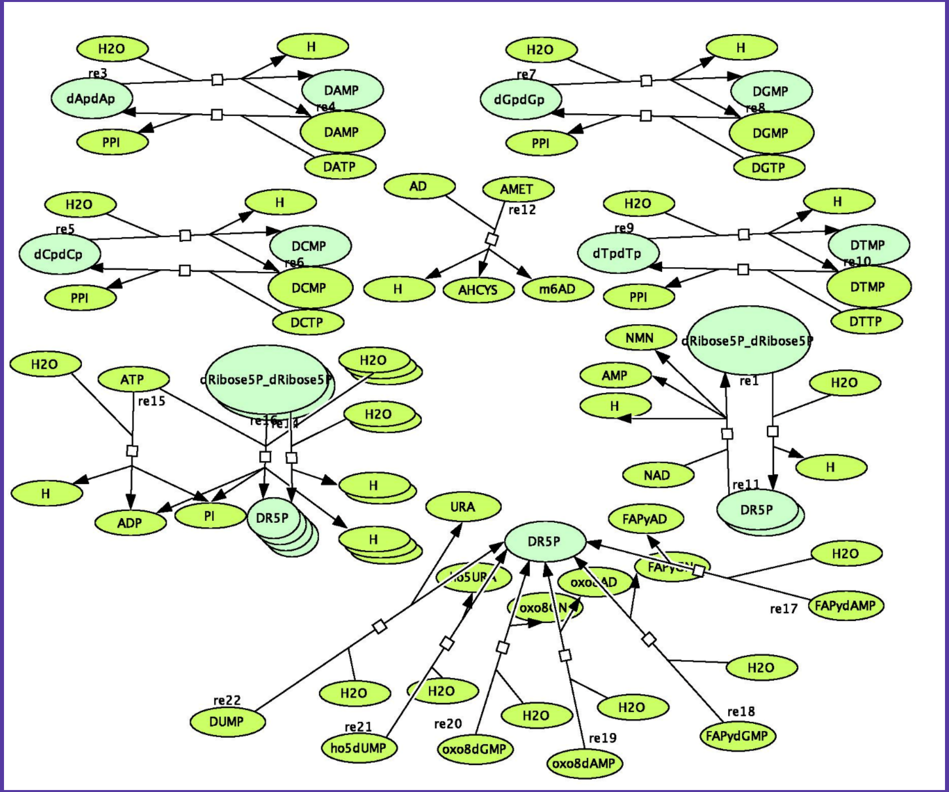
So far…

- Extracted information associated with DNA Damage and DNA Repair modules from Table S3.
- Arne developed a matlab code to extract the information from the table.
- Generated a template SBML and SBGN map for both the modules.

SBGN

DNA Repair module – SBGN map

Working on fixing errors



Things not clear…

- Values of variables and parameters, and i/os not clear
- Poisson distribution (random numbers) – SBML array and distrib package – What tools handle these packages?
- SED-ML – sequential simulation

More to learn… during the worksho

Work update – 10th March 2015

- SBGN generation (Audald, Namrata, Mahesh)
  - Imported the already generated SBGN diagram using VANTED. The diagrams are close to final. We need to create clone for some species to make a neat figure.
- SBML encode (Audald, Namrata, Mahesh)
  - Encoded both the modules using Copasi. The kinetic laws are not added yet.
- Matlab code – Input/Outputs, running submodule (Arne, Viji)
  - Looking into the matlab code for providing information to the integration team.
  - Looked into running the submodules separately

Work update – 11th March 2015

- SBGN generation (Namrata, Mahesh)
  - Finalize the map
- Interface
  - Input/Outputs (Audald)
  - Matlab code – extracting information (Arne)
- SBML encode (working on it)
  - Chromosome position
  - randomness

Work update – 12th March 2015

- SBGN generation (Namrata, Mahesh, Audald)
  - Finalize the DNA damage and repair map
  - Input/Outputs (Audald)
  - Matlab code – extracting information (Arne)
- SBML encode (working on it)
  - Chromosome position
  - randomness

Work update – 13th March 2015

- SBGN generation
  - Finalized the map
- Interface
  - Input/Outputs (physical entities and parameters) - extracted from matlab code
- Encoded SBML files (has done top level annotation) for the both the modules, but still working on
  - Chromosome position
  - randomness

32 reactions (out of 140 in knowledge base)

DNA Damage

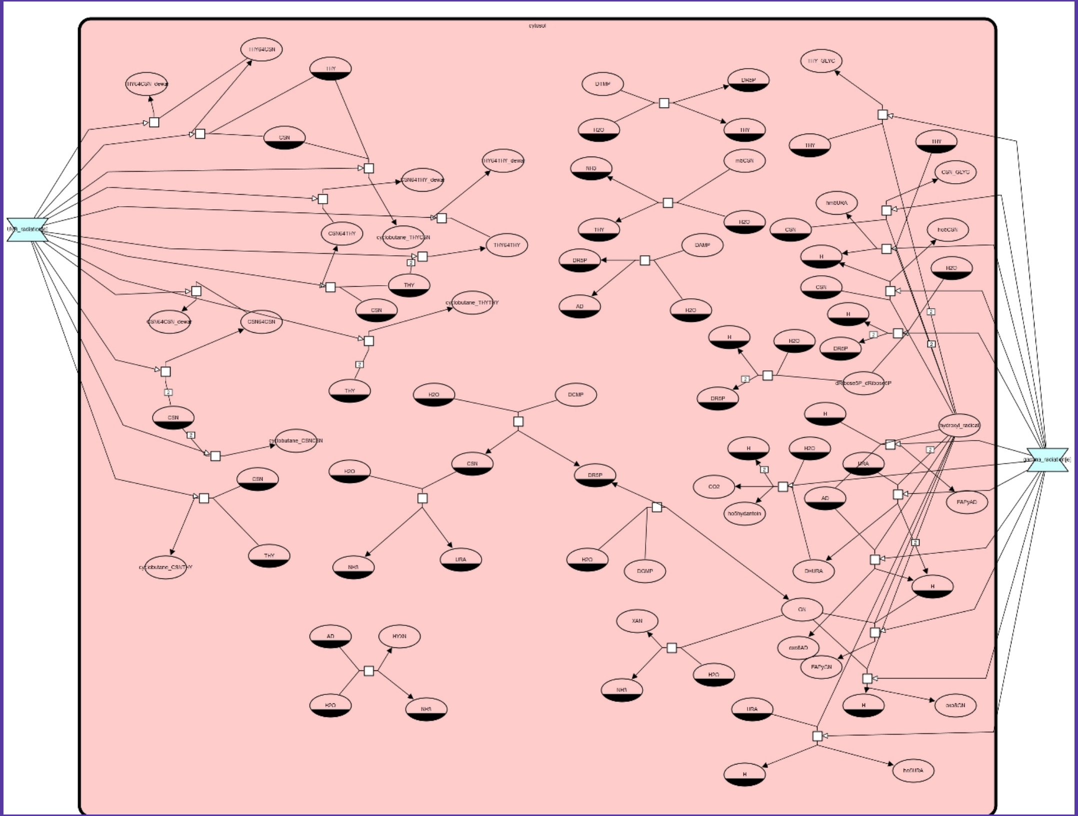
UV-B photodimerization

radiation (gamma-ray) induced base oxidation

spontaneous base deamination

spontaneous base loss

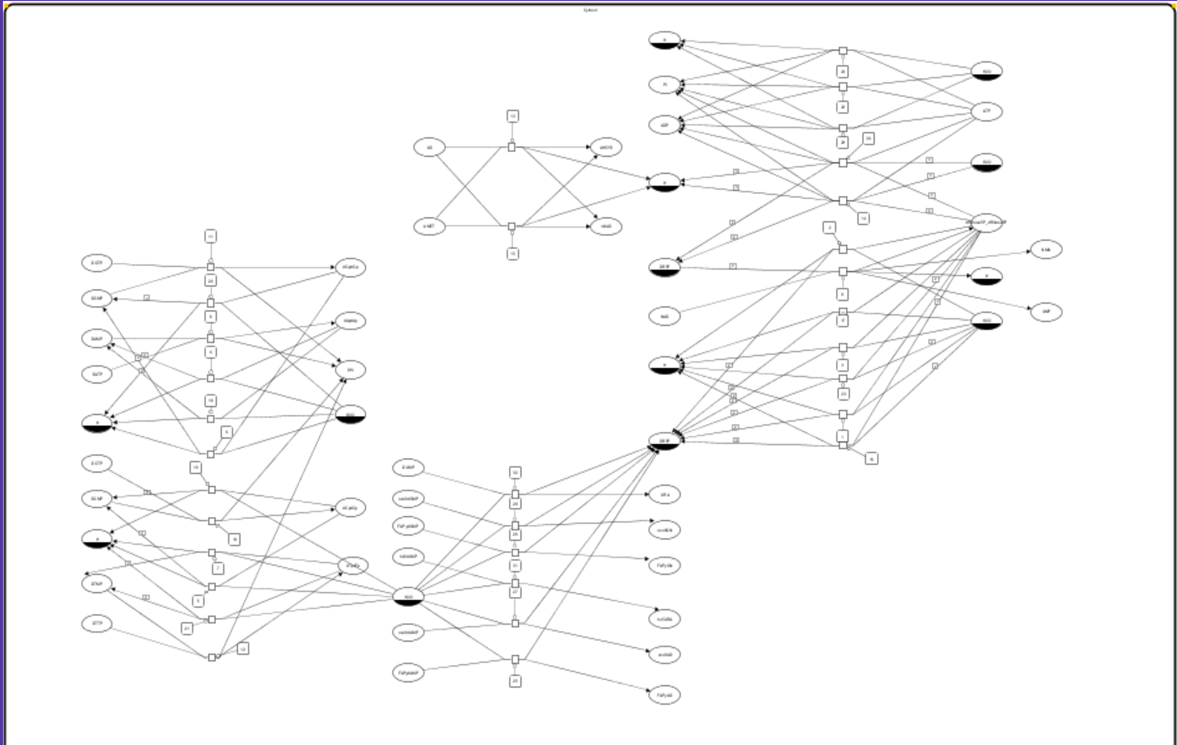
strand break



32 reactions involving 15 enzymes:

DNA Repair

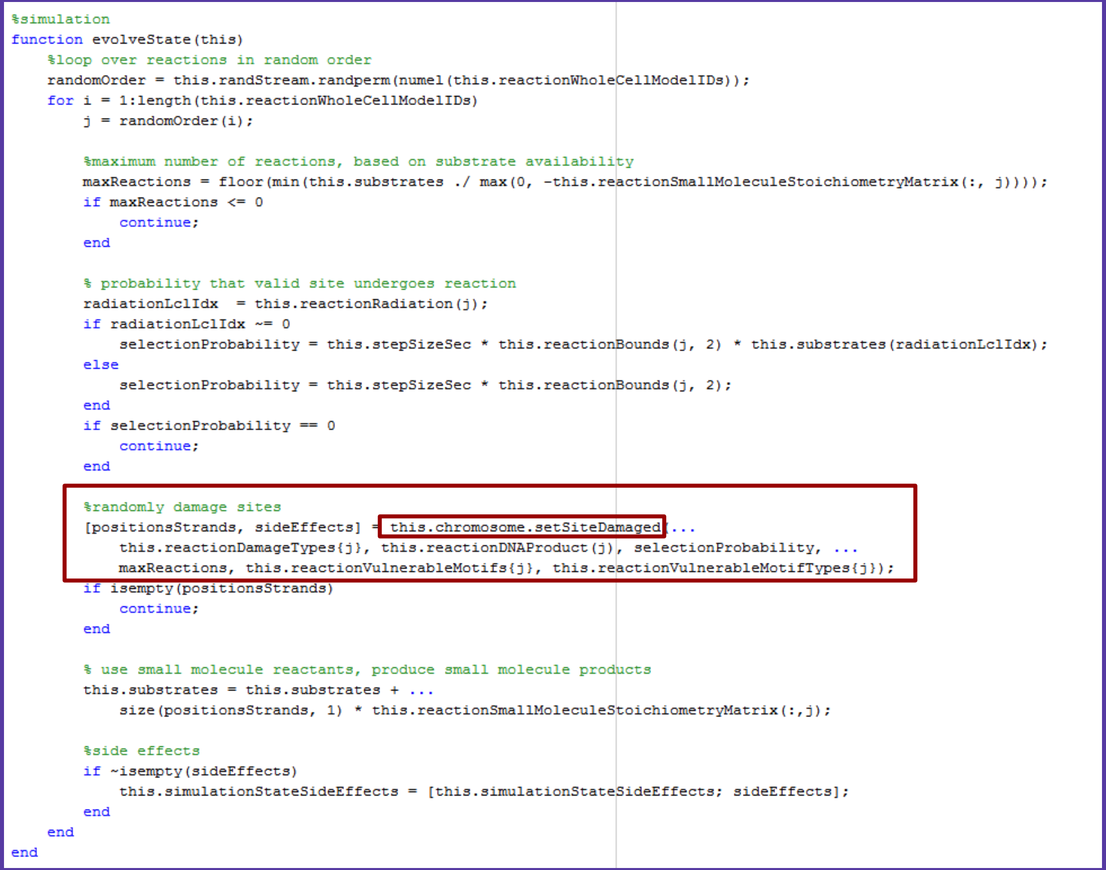
- Apurinic endonuclease
- Formamidopyrimidine-DNA glycosylase
- Phosphoesterase
- DNA ligase, NAD-dependent
- DNA polymerase III, beta subunit
- Type I restriction modification DNA specificity domain protein
- PolI-like 5'-3' exonuclease, putative
- Recombination protein, strand exchange
- Uracil-DNA glycosylase, putative
- Adenine-specific DNA modification methylase
- Holliday junction DNA helicase
- Holliday junction endonuclease
- 3-5' endonuclease
- DNA incision complex
- Putative DNA integrity scanning protein



DNA Damage

Main work is done in Chromosome.m

- changes substrate (small molecule) concentrations

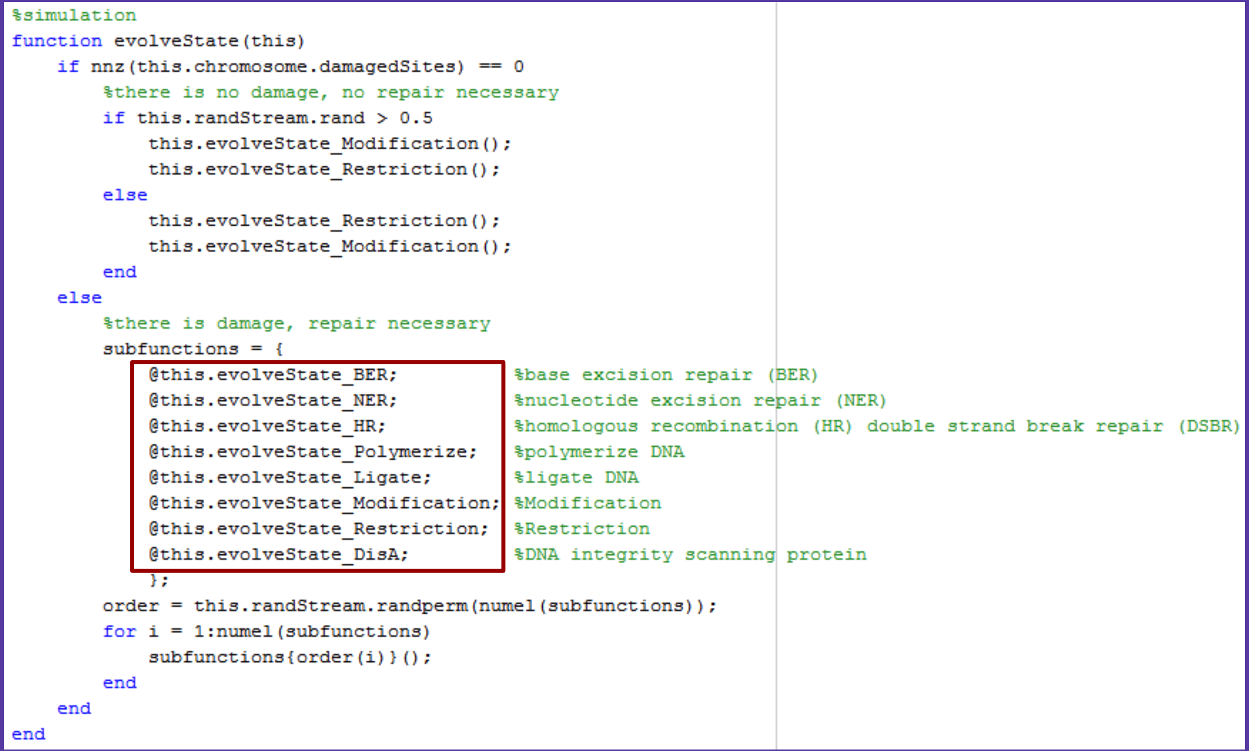


```
%simulation
function evolveState(this)
    %loop over reactions in random order
    randomOrder = this.randStream.randperm(numel(this.reactionWholeCellModelIDs));
    for i = 1:length(this.reactionWholeCellModelIDs)
        j = randomOrder(i);

        %maximum number of reactions, based on substrate availability
        maxReactions = floor(min(this.substrates ./ max(0, -this.reactionSmallMoleculeStoichiometryMatrix(:, j))));
        if maxReactions <= 0
            continue;
        end

        % probability that valid site undergoes reaction
        radiationLclIdx  = this.reactionRadiation(j);
        if radiationLclIdx ~= 0
            selectionProbability = this.stepSizeSec * this.reactionBounds(j, 2) * this.substrates(radiationLclIdx);
        else
            selectionProbability = this.stepSizeSec * this.reactionBounds(j, 2);
        end
        if selectionProbability == 0
            continue;
        end

        %randomly damage sites
        [positionsStrands, sideEffects] = this.chromosome.setSiteDamaged(...
            this.reactionDamageTypes{j}, this.reactionDNAProduct(j), selectionProbability, ...
            maxReactions, this.reactionVulnerableMotifs{j}, this.reactionVulnerableMotifTypes{j});
        if isempty(positionsStrands)
            continue;
        end

        % use small molecule reactants, produce small molecule products
        this.substrates = this.substrates + ...
            size(positionsStrands, 1) * this.reactionSmallMoleculeStoichiometryMatrix(:,j);

        %side effects
        if ~isempty(sideEffects)
            this.simulationStateSideEffects = [this.simulationStateSideEffects; sideEffects];
        end
    end
end
```

DNA Repair

basically undoes DNA Damage

...via different (more complex) functions



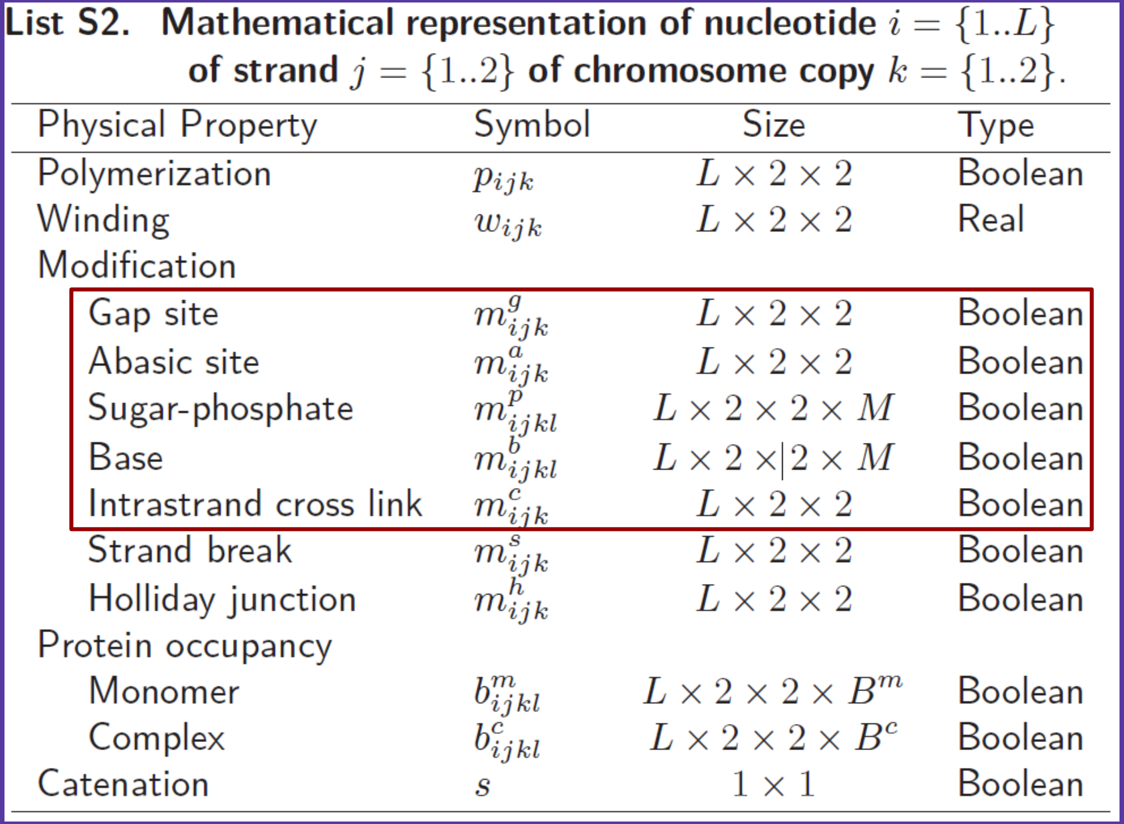
```
%simulation
function evolveState(this)
    if nnz(this.chromosome.damagedSites) == 0
        %there is no damage, no repair necessary
        if this.randStream.rand > 0.5
            this.evolveState_Modification();
            this.evolveState_Restriction();
        else
            this.evolveState_Restriction();
            this.evolveState_Modification();
        end
    else
        %there is damage, repair necessary
        subfunctions = {
            @this.evolveState_BER;           %base excision repair (BER)
            @this.evolveState_NER;           %nucleotide excision repair (NER)
            @this.evolveState_HR;            %homologous recombination (HR) double strand break repair (DSBR)
            @this.evolveState_Polymerize;    %polymerize DNA
            @this.evolveState_Ligate;        %ligate DNA
            @this.evolveState_Modification;  %Modification
            @this.evolveState_Restriction;   %Restriction
            @this.evolveState_DisA;          %DNA integrity scanning protein
            };
        order = this.randStream.randperm(numel(subfunctions));
        for i = 1:numel(subfunctions)
            subfunctions{order(i)}();
        end
    end
end
```

Shared Chromosome State

Chromosome class encapsulates modifications and their effects

Do you know how changes here affect your model?

Chromosome representation is crucial



List S2. Mathematical representation of nucleotide $i = \{1..L\}$ of strand $j = \{1..2\}$ of chromosome copy $k = \{1..2\}$.

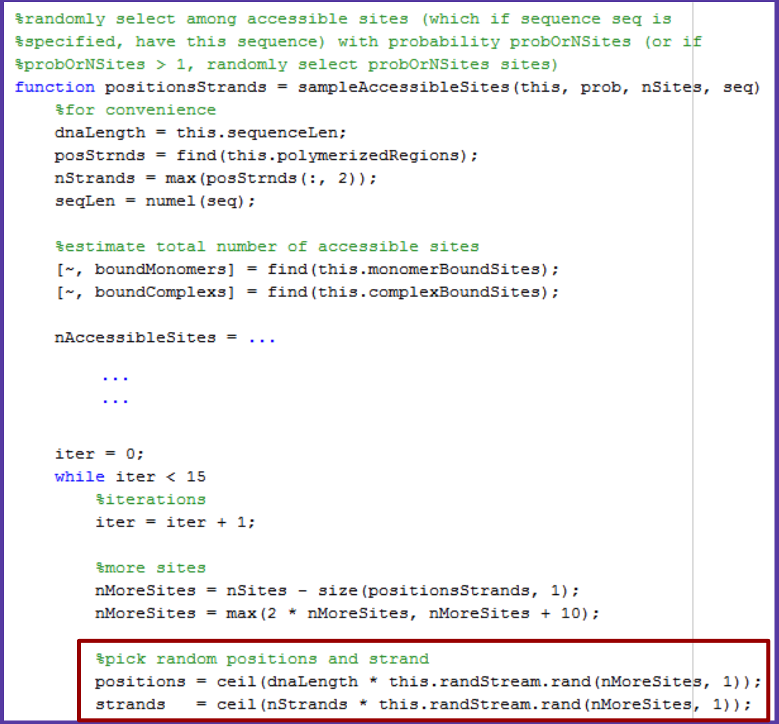
Physical Property	Symbol	Size	Type
Polymerization	p_{ijk}	$L \times 2 \times 2$	Boolean
Winding	w_{ijk}	$L \times 2 \times 2$	Real
Modification			
Gap site	m^g_{ijk}	$L \times 2 \times 2$	Boolean
Abasic site	m^a_{ijk}	$L \times 2 \times 2$	Boolean
Sugar-phosphate	m^p_{ijkl}	$L \times 2 \times 2 \times M$	Boolean
Base	m^b_{ijkl}	$L \times 2 \times 2 \times M$	Boolean
Intrastrand cross link	m^c_{ijk}	$L \times 2 \times 2$	Boolean
Strand break	m^s_{ijk}	$L \times 2 \times 2$	Boolean
Holliday junction	m^h_{ijk}	$L \times 2 \times 2$	Boolean
Protein occupancy			
Monomer	b^m_{ijkl}	$L \times 2 \times 2 \times B^m$	Boolean
Complex	b^c_{ijkl}	$L \times 2 \times 2 \times B^c$	Boolean
Catenation	s	1×1	Boolean

Randomness

...comes in at modification site selection and reaction order

Our reactions change metabolites, but not the chromosomes so far

Execution order randomization may be artifact of split into submodules



```
%randomly select among accessible sites (which if sequence seq is
%specified, have this sequence) with probability probOrNSites (or if
%probOrNSites > 1, randomly select probOrNSites sites)
function positionsStrands = sampleAccessibleSites(this, prob, nSites, seq)
    %for convenience
    dnaLength = this.sequenceLen;
    posStrnds = find(this.polymerizedRegions);
    nStrands = max(posStrnds(:, 2));
    seqLen = numel(seq);

    %estimate total number of accessible sites
    [~, boundMonomers] = find(this.monomerBoundSites);
    [~, boundComplexs] = find(this.complexBoundSites);

    nAccessibleSites = ...

        ...
        ...

    iter = 0;
    while iter < 15
        %iterations
        iter = iter + 1;

        %more sites
        nMoreSites = nSites - size(positionsStrands, 1);
        nMoreSites = max(2 * nMoreSites, nMoreSites + 10);

        %pick random positions and strand
        positions = ceil(dnaLength * this.randStream.rand(nMoreSites, 1));
        strands   = ceil(nStrands * this.randStream.rand(nMoreSites, 1));
```