Testing chemical and biochemical fidelity

Authors: Ronan Fleming, Ines Thiele, University of Luxembourg.

Reviewer: Almut Heinken, University of Luxembourg.

Introduction

Once a context-specific model is generated, but before a it is used to make predictions of biological relevance, it should be subjected to a range of quantitative and qualitative chemical and biochemical fidelity tests. The stoichiometric consistency tests should not be necessary if one starts with a generic model where the internal reactions are all stoichiometrically consistent then a context-specific model extracted from it should also be stoichiometrically consistent. Beyond chemical fidelity, it is also very important to test biochemical fidelity. Such tests are very specific to the particular biological domain one is modelling. Here we focus on human metabolism and use the Recon3.0model or Recon2.0 model.

PROCEDURE

Load a model

Load Recon3.0model, or use Recon2.0model instead. You may also load your own model.

```
clear %model
if exist('2017_04_28_Recon3dForCurrentDistribution.mat','file')==2
    filename = '2017_04_28_Recon3dForCurrentDistribution.mat';
    load(filename);
    model=modelRecon3model;
    clear modelRecon3model;
    model.csense(1:size(model.S,1),1)='E';
    if ~isfield(model,'modelID')
        model.modelID='Recon3.0model';
    end
else
    filename2='Recon2.0model.mat';
    if exist('Recon2.0model.mat','file')==2
        load(filename2);
        model=Recon2model;
        clear Recon2model;
        model.csense(1:size(model.S,1),1)='E';
        if ~isfield(model,'modelID')
        model.modelID='Recon2.0model';
    end
    end
end
```

Display the size of the model

```
[nMet,nRxn] = size(model.S);
fprintf('%6s\t%6s\n','#mets','#rxns'); fprintf('%6u\t%6u\t%s%s\n',nMet,nRxn,' totals in ', mod
```

```
  #mets  #rxns
   5835  10600  totals in Recon3.0model
```

```

```

Set the threshold to classify flux into non-zero and zero flux:

```
threshold=1e-6;
```

Set a solver

```
% changeCobraSolver ('gurobi', 'all', 1);
changeCobraSolver ('glpk', 'all', 1);
```

```
> Solver for LPproblems has been set to glpk.
> Solver for MILPproblems has been set to glpk.
> Solver glpk not supported for problems of type MIQP. Currently used: tomlab_cplex
> Solver glpk not supported for problems of type NLP. Currently used: tomlab_snopt
> Solver glpk not supported for problems of type QP. Currently used: qpng
```

Production of mthgxl from 12ppd-S

Add sink reactions for either end of the proposed pathway:

```
model = addSinkReactions(model,{'12ppd_S[c]','mthgxl[c]'},[-100 -1; 0 100]);
```

```
sink_12ppd_S[c] 12ppd_S[c]  <=>
sink_mthgxl[c] mthgxl[c]  ->
```

Change the objective to maximise the sink reaction for mthgxl[c]

```
model = changeObjective(model,'sink_mthgxl[c]');
```

Test if it is possible to attain a nonzero objective, and if it is compute a sparse flux vector:

```
sol = optimizeCbModel(model,'max','zero');
```

Check to see if there is a non-zero flux through the objective

```
if sol.stat==1
    fprintf('%g%s\n',sol.v(model.c~=0),' flux through the sink_mthgxl[c] reaction')
end
```

```
100 flux through the sink_mthgxl[c] reaction
```

Display the sparse flux solution, but only the non-zero fluxes, above a specified threshold.

```
if sol.stat==1
    for n=1:nRxn
        if abs(sol.v(n))>threshold
            formula=printRxnFormula(model, model.rxns{n}, 0);
            fprintf('%10g%15s\t%-60s\n',sol.v(n),model.rxns{n}, formula{1});
        end
    end
end
```

```
         1       ALCD21_L nad[c] + 12ppd_S[c]  -> h[c] + nadh[c] + lald_L[c]
         1       ALCD22_L nad[c] + lald_L[c]  -> h[c] + nadh[c] + mthgxl[c]
```

```
 0.5          FOLt2 h2o[c] + fol[e]  <=> h2o[e] + fol[c]
49.5           MGSA dhap[c]  -> pi[c] + mthgxl[c]
49.5          MGSA2 g3p[c]  -> pi[c] + mthgxl[c]
 0.5          r0224 h[c] + nadh[c] + dhf[c]  -> nad[c] + thf[c]
-0.5          r0512 nad[c] + dhf[c]  <=> nadh[c] + fol[c]
-0.5          r0688 h2o[c] + nad[c] + xol7ah2al[c]  <=> 2 h[c] + nadh[c] + dhcholestanate[c]
-0.5          r0739 nad[c] + xol7ah3[c]  <=> h[c] + nadh[c] + xol7ah2al[c]
  99          r1423 pi[c]  -> pi[e]
 0.5    sink_thf[c] thf[c]  <=>
-0.5sink_dhcholestanate[c] dhcholestanate[c]  <=>
 0.5sink_xol7ah3[c] xol7ah3[c]  <=>
 0.5      EX_h2o[e] h2o[e]  <=>
  99       EX_pi[e] pi[e]  <=>
49.5            FBA fdp[c]  <=> dhap[c] + g3p[c]
-0.5      EX_fol[e] fol[e]  <=>
-49.5          FDPte fdp[c]  <=> fdp[e]
-49.5      EX_fdp[e] fdp[e]  <=>
```

ANTICIPATED RESULTS

If FBAsol.stat==1 then it is feasible to produce methylglyoxal from (S)-propane-1,2-diol. If FBAsol.stat==0, then this metabolic function is infeasible. This is not anticipated and indicates that further gap filling is required (cf Gap Filling Tutorial).

Metabolic task: 4abut -> succ[m]

Add sink reactions for either end of the proposed pathway:

```
model = addSinkReactions(model,{'4abut[c]','succ[m]'},[-100 -1; 0 100]);
```

```
sink_4abut[c] 4abut[c]  <=>
sink_succ[m] succ[m]  ->
```

Change the objective to maximise the sink reaction for nh4[c]

```
model = changeObjective(model,'sink_succ[m]');
```

Test if it is possible to attain a nonzero objective, and if it is compute a sparse flux vector:

```
sol = optimizeCbModel(model,'max','zero');
```

Check to see if there is a non-zero flux through the objective

```
if sol.stat==1
    fprintf('%g%s\n',sol.v(model.c~=0),' flux through the sink_succ[m] reaction')
end
```

```
100 flux through the sink_succ[m] reaction
```

Display the sparse flux solution, but only the non-zero fluxes, above a specified threshold.

```
if sol.stat==1
```

```
    for n=1:nRxn
        if abs(sol.v(n))>threshold
            formula=printRxnFormula(model, model.rxns{n}, 0);
            fprintf('%10g%15s\t%-60s\n',sol.v(n),model.rxns{n}, formula{1});
        end
    end
end
```

```
  0.117014         3AIBTm 2mop[m] + glu_L[m]  <=> akg[m] + 3aib[m]
  0.117014        3AIBtmi 3aib[m]  -> 3aib[c]
    5.4072          ADK1m atp[m] + amp[m]  <=> 2 adp[m]
         1       ALCD21_L nad[c] + 12ppd_S[c]  -> h[c] + nadh[c] + lald_L[c]
  0.940148       ALCD22_L nad[c] + lald_L[c]  -> h[c] + nadh[c] + mthgxl[c]
   1.64223         ARGDCm h[m] + arg_L[m]  -> co2[m] + agm[m]
   1.64223          ARGtm h[m] + arg_L[c]  <=> h[c] + arg_L[m]
 0.0585071          ASNNm h2o[m] + asn_L[m]  -> asp_L[m] + nh4[m]
 0.0585071          ASNtm asn_L[c]  -> asn_L[m]
 0.0585071         ASPTAm akg[m] + asp_L[m]  <=> glu_L[m] + oaa[m]
 0.0598521          CO2tm co2[c]  <=> co2[m]
-0.940148        D_LACtm h[c] + lac_D[c]  <=> h[m] + lac_D[m]
  0.117014     EX_3aib[e] 3aib[e]  <=>
  -3.82044      EX_utp[e] utp[e]  <=>
  -1.76059           FUMm h2o[m] + fum[m]  <=> mal_L[m]
   23.4452          FUMtm pi[m] + fum[c]  <=> pi[c] + fum[m]
  0.940148           GGNG Tyr_ggn[c] + 8 udpg[c]  -> 8 h[c] + 8 udp[c] + ggn[c]
  0.940148         GLBRAN glygn1[c]  -> glygn2[c]
  0.940148         GLGNS1 3 udpg[c] + ggn[c]  -> 3 h[c] + 3 udp[c] + glygn1[c]
   3.44721        GLPASE1 3 pi[c] + glygn2[c]  -> 3 g1p[c] + dxtrn[c]
-0.0585071         GLUDxm h2o[m] + nad[m] + glu_L[m]  <=> h[m] + akg[m] + nadh[m] + nh4[m]
  0.940148         GLYOXm h2o[m] + lgt_S[m]  -> h[m] + lac_D[m] + gthrd[m]
   1.82179         GTHRDt h2o[c] + atp[c] + gthrd[c]  -> h[c] + adp[c] + pi[c] + gthrd[m]
   1.70208        H2CO3Dm h2o[m] + co2[m]  -> h[m] + hco3[m]
  -1.76059           MDHm nad[m] + mal_L[m]  <=> h[m] + nadh[m] + oaa[m]
 0.0598521            ME2 nadp[c] + mal_L[c]  -> pyr[c] + nadph[c] + co2[c]
  -12.0464          MMEm mmcoa_R[m]  <=> mmcoa_S[m]
   12.0464          MMMm mmcoa_R[m]  <=> succoa[m]
    36.076          O2tm o2[c]  <=> o2[m]
   1.70208           PCm pyr[m] + atp[m] + hco3[m]  -> h[m] + adp[m] + pi[m] + oaa[m]
   15.7488          PPAm h2o[m] + ppi[m]  -> h[m] + 2 pi[m]
   1.70208        PYRt2m h[c] + pyr[c]  -> h[m] + pyr[m]
  -25.2058         SUCD1m fad[m] + succ[m]  <=> fadh2[m] + fum[m]
  -12.0464        SUCOASm coa[m] + atp[m] + succ[m]  <=> adp[m] + pi[m] + succoa[m]
  -3.82044          UMPK3 utp[c] + ump[c]  <=> 2 udp[c]
  -1.85104           URIt uri[e]  <=> uri[c]
  0.702085          r0165 h[c] + udp[c] + pep[c]  -> pyr[c] + utp[c]
    37.542          r0178 h2o[m] + nad[m] + sucsal[m]  <=> 2 h[m] + nadh[m] + succ[m]
   25.2058          r0179 h2o[m] + nadp[m] + sucsal[m]  -> 2 h[m] + nadph[m] + succ[m]
  -12.0464          r0571 h2o[m] + mmcoa_S[m]  <=> h[m] + coa[m] + HC00900[m]
  -35.6059          r0616 nad[m] + 4hpro_LT[m]  <=> 2 h[m] + nadh[m] + 1p3h5c[m]
 -0.117014          r0643 h2o[m] + nad[m] + 2mop[m]  <=> 2 h[m] + nadh[m] + HC00900[m]
  -25.2058          r0652 nadp[m] + tdcoa[m]  <=> h[m] + nadph[m] + HC01412[m]
  -2.76194          r0885 pi[m] + gthrd[c]  <=> pi[c] + gthrd[m]
  -3.82044          r0892 utp[c]  <=> utp[e]
  -3.82044          r1156 uri[c] + dutp[c]  <=> h[c] + dudp[c] + ump[c]
```

```
  0.234028      3HCO3_NAt 3 hco3[e] + na1[e]  <=> 3 hco3[c] + na1[c]
   10.3416          PPItm ppi[c]  <=> ppi[m]
  -0.702085          PEPtr hco3[e] + pep[c]  <=> hco3[c] + pep[e]
   21.5649          FUMtr fum[e]  <=> fum[c]
  -0.702085      EX_pep[e] pep[e]  <=>
   -12.1634  EX_HC00900[e] HC00900[e]  <=>
   -12.1634      HC00900t4 pi[e] + HC00900[c]  <=> pi[c] + HC00900[e]
   -1.94015           OAAt h[c] + oaa[c]  <=> h[e] + oaa[e]
   -1.94015      EX_oaa[e] oaa[e]  <=>
 -0.0292535EX_asnasnarg[e] asnasnarg[e]  <=>
  0.0292535     ASNASNARGt h[e] + asnasnarg[e]  <=> h[c] + asnasnarg[c]
  0.0292535     ASNASNARGr 2 h2o[c] + asnasnarg[c]  <=> 2 asn_L[c] + arg_L[c]
   -12.1634         MMALtm HC00900[m]  <=> HC00900[c]
 -0.940148sink_Tyr_ggn[c] Tyr_ggn[c]  <=>
  -2.50706 sink_glygn2[c] glygn2[c]  <=>
   3.44721         DXTRNt dxtrn[c]  <=> dxtrn[e]
   3.44721    EX_dxtrn[e] dxtrn[e]  <=>
   -21.5649      EX_fum[e] fum[e]  <=>
    -36.076       EX_o2[e] o2[e]  <=>
    12.1634       EX_pi[e] pi[e]  <=>
   3.82044      EX_uri[e] uri[e]  <=>
    -1.8803            FUM h2o[c] + fum[c]  <=> mal_L[c]
    10.3416           GALU h[c] + g1p[c] + utp[c]  <=> ppi[c] + udpg[c]
    1.99866          NDPK2 atp[c] + udp[c]  <=> adp[c] + utp[c]
   -3.82044          NDPK6 atp[c] + dudp[c]  <=> adp[c] + dutp[c]
     36.076            O2t o2[e]  <=> o2[c]
    -1.9694         URIt2r h[e] + uri[e]  <=> h[c] + uri[c]
   0.940148          LGTHL mthgxl[c] + gthrd[c]  -> lgt_S[c]
   -1.94015            MDH nad[c] + mal_L[c]  <=> h[c] + nadh[c] + oaa[c]
  -0.117014      3AIBSYMPt 2 na1[e] + 3aib[e]  <=> 2 na1[c] + 3aib[c]
    1.64223         AGRMte agm[c]  <=> agm[e]
    1.64223         AGRMtm agm[m]  <=> agm[c]
    1.64223      EX_agm[e] agm[e]  <=>
   -62.7478       SUCSALtm sucsal[m]  <=> sucsal[c]
   -62.7478       SUCSALte sucsal[c]  <=> sucsal[e]
   -62.7478   EX_sucsal[e] sucsal[e]  <=>
  0.0598521       12PPDRte 12ppd_R[c]  <=> 12ppd_R[e]
  0.0598521  EX_12ppd_R[e] 12ppd_R[e]  <=>
   25.2058       HMR_3128 fad[m] + tdcoa[m]  -> fadh2[m] + HC01412[m]
  0.0598521       HMR_3855 h[c] + nadph[c] + lald_L[c]  -> nadp[c] + 12ppd_R[c]
   0.940148       HMR_3859 2 ficytC[m] + lac_D[c]  -> 2 h[c] + pyr[c] + 2 focytC[m]
     5.4072       HMR_3966 h2o[m] + atp[m]  -> h[m] + amp[m] + ppi[m]
   35.6059       HMR_4783 o2[m] + h[m] + 4hpro_LT[m]  -> 2 h2o[m] + 1p3h5c[m]
   0.940148       HMR_8510 lgt_S[c]  <=> lgt_S[m]
   -1.61298  sink_arg_L[c] arg_L[c]  <=>
          1    DM_4abut[c] 4abut[c]  ->
   0.470074        ATPS4mi adp[m] + pi[m] + 4 h[i]  -> h2o[m] + 3 h[m] + atp[m]
   0.470074        CYOOm2i o2[m] + 8 h[m] + 4 focytC[m]  -> 2 h2o[m] + 4 ficytC[m] + 4 h[i]
```

ANTICIPATED RESULTS

If FBAsol.stat==1 then it is feasible to produce mitochondrial succinate from 4-Aminobutanoate. If FBAsol.stat==0, then this metabolic function is infeasible. This is not anticipated and indicates that further gap filling is required (cf Gap Filling Tutorial).

Metabolic task: gly -> co2 and nh4 (via glycine cleavage system)

Add sink reactions for either end of the proposed pathway:

```
model = addSinkReactions(model,{'gly[c]','co2[c]','nh4[c]'},[-100 -1; 0.1 100; 0.1 100]);
```

```
Warning: Reaction with the same name already exists in the model, updating the reaction
sink_gly[c] gly[c]  <=>
sink_co2[c] co2[c]  ->
sink_nh4[c] nh4[c]  ->
```

Change the objective to maximise the sink reaction for nh4[c]

```
model = changeObjective(model,'sink_nh4[c]');
```

Test if it is possible to attain a nonzero objective, and if it is compute a sparse flux vector:

```
sol = optimizeCbModel(model,'max','zero');
```

Check to see if there is a non-zero flux through the objective

```
if sol.stat==1
    fprintf('%g%s\n',sol.v(model.c~=0),' flux through the sink_nh4[c] reaction')
end
```

```
100 flux through the sink_nh4[c] reaction
```

Display the sparse flux solution, but only the non-zero fluxes, above a specified threshold.

```
if sol.stat==1
    for n=1:nRxn
        if abs(sol.v(n))>threshold
            formula=printRxnFormula(model, model.rxns{n}, 0);
            fprintf('%10g%15s\t%-60s\n',sol.v(n),model.rxns{n}, formula{1});
        end
    end
end
```

```
         1        AKR1C41 h[c] + nadh[c] + xol7ah[c]  -> nad[c] + xol7ah2[c]
         1       ALCD21_L nad[c] + 12ppd_S[c]  -> h[c] + nadh[c] + lald_L[c]
 0.0333333          GLUDC h[c] + glu_L[c]  -> co2[c] + 4abut[c]
         1          r0747 nadp[c] + xol7ah2[c]  <=> h[c] + nadph[c] + xol7ah[c]
  -0.12381          r1088 h[e] + cit[e]  <=> h[c] + cit[c]
 0.0333333          r1702 na1[e] + gln_L[e] + gly[c]  -> na1[c] + gln_L[c] + gly[e]
 0.0166667          r2008 gly[c] + arg_L[e]  -> gly[e] + arg_L[c]
 0.0166667        RE3052C cpppg3[c]  -> 6 h[c] + C05770[c]
   0.12381        CITt4_4 4 na1[e] + cit[e]  <=> 4 na1[c] + cit[c]
-0.0166667      GLYGLYCNc h2o[c] + glygly[c]  <=> 2 gly[c]
 -0.528571     GLYSNAT5tc h[c] + na1[e] + gly[e]  <=> h[e] + na1[c] + gly[c]
```

```
 0.0166667EX_argglygly[e] argglygly[e]  <=>
-0.0166667     ARGGLYGLYt h[e] + argglygly[e]  <=> h[c] + argglygly[c]
 0.0166667     ARGGLYGLYr arg_L[c] + glygly[c]  <=> h2o[c] + argglygly[c]
 0.0166667           HMBS h2o[c] + 4 ppbng[c]  -> 4 nh4[c] + hmbil[c]
 0.0166667          UPP3S hmbil[c]  -> h2o[c] + uppg3[c]
 0.0166667         UPPDC1 4 h[c] + uppg3[c]  -> 4 co2[c] + cpppg3[c]
  0.966667      EX_gly[e] gly[e]  <=>
 -0.388095         GLYt2r h[e] + gly[e]  <=> h[c] + gly[c]
-0.0333333    EX_gln_L[e] gln_L[e]  <=>
-0.0166667    EX_arg_L[e] arg_L[e]  <=>
     -99.9      EX_nh4[e] nh4[e]  <=>
      99.9          NH4tb nh4[e]  <=> nh4[c]
 0.0166667      C05770te3 C05770[c]  -> C05770[e]
 0.0166667   EX_C05770[e] C05770[e]  <=>
-0.0666667        PPBNGte ppbng[c]  <=> ppbng[e]
-0.0666667    EX_ppbng[e] ppbng[e]  <=>
         1       12PPDRte 12ppd_R[c]  <=> 12ppd_R[e]
         1  EX_12ppd_R[e] 12ppd_R[e]  <=>
         1       HMR_3855 h[c] + nadph[c] + lald_L[c]  -> nadp[c] + 12ppd_R[c]
 0.0333333       HMR_9802 h2o[c] + gln_L[c]  -> nh4[c] + glu_L[c]
        -1    sink_gly[c] gly[c]  <=>
   1.03333    DM_4abut[c] 4abut[c]  ->
```

ANTICIPATED RESULTS

If FBAsol.stat==1 then it is feasible to produce CO2 and NH4 from glycine. If FBAsol.stat==0, then this metabolic function is infeasible. This is not anticipated and indicates that further gap filling is required (cf Gap Filling Tutorial).

REFERENCES

[fleming_cardinality_nodate] Fleming, R.M.T., et al., Cardinality optimisation in constraint-based modelling: illustration with Recon 3D (submitted), 2017.

[sparsePaper] Le Thi, H.A., Pham Dinh, T., Le, H.M., and Vo, X.T. (2015). DC approximation approaches for sparse optimization. European Journal of Operational Research 244, 26–46.