Technical report 2009-01

Timur Fayruzov

1 Fission Yeast network model

The network that models the fission yeast cell cycle is described in [2] and is shown in Figure 1. We take this network and translate it to our framework. The description of a general framework, in lparse syntax is presented below
#option –true-negation.
#domain time(T), index(J), index(J1), index_0(A), index_0(I), thresh(Th).
#const n=10.
#const neg_n=-1.

$index(1..n).$
$index\_0(0..n).$
$thresh(neg\_n..1).$

rule $G1$: $time(0..0).$

rule $G2$: $act(Y, T+1) \leftarrow not\ conflict(Y,T), not\ mod\_act\_th(Y), act(X,T), activates(X,Y,T).$

rule $G2.1$: $act(Y, T+1) \leftarrow conflict(Y,T), act\_th(Y,Th), prot(Y), \#\_act(Y,A,T), \#\_inh(Y,I,T), A - I > Th.$

rule $G2.2$: $act(Y, T+1) \leftarrow not\ conflict(Y,T), prot(Y), act\_th(Y,Th), Th! = 0, \#\_act(Y,A,T), \#\_inh(Y,I,T), A - I > Th.$

rule $G3$: $inh(Y, T+1) \leftarrow not\ conflict(Y,T), not\ mod\_act\_th(Y), act(X,T), inhibits(X,Y,T).$

rule $G3.1$: $inh(Y, T+1) \leftarrow conflict(Y,T), inh\_th(Y,Th), prot(Y), \#\_act(Y,A,T), \#\_inh(Y,I,T), I - A > Th.$

rule $G3.2$: $inh(Y, T+1) \leftarrow not\ conflict(Y,T), prot(Y), inh\_th(Y,Th), Th! = 0, \#\_inh(Y,I,T), \#\_act(Y,A,T), I - A > Th.$

rule $G4$: $conflict(X,T) \leftarrow activates(Y,X,T), act(Y,T), inhibits(Z,X,T), act(Z,T).$

rule $G5$: $act(X, T+1) \leftarrow act(X,T), not\ inh(X, T+1), prot(X).$

rule $G6$: $inh(X, T+1) \leftarrow inh(X,T), not\ act(X, T+1), prot(X).$

rule $G7$: $act\_th(X, 0) \leftarrow prot(X), not\ mod\_act\_th(X).$

rule $G7.1$: $mod\_act\_th(X) \leftarrow act\_th(X,Th), Th! = 0, prot(X).$

rule $G8$: $inh\_th(X, 0) \leftarrow prot(X), not\ mod\_inh\_th(X).$

rule $G8.1$: $mod\_inh\_th(X) \leftarrow inh\_th(X,Th), Th! = 0, prot(X).$

rule $G9$: $act(X, 0) \leftarrow not\ inh(X,0), prot(X).$

rule $G10$: $inh(X, 0) \leftarrow not\ act(X,0), prot(X).$

The specific part that describes a structure of the network looks as follows

$prot(sk).$
$prot(ste9).$
$prot(rum1).$
$prot(cdc2Cdc13).$
$prot(cdc25).$
$prot(pp).$
$prot(wee1Mik1).$
$prot(slp1).$
$prot(cdc2Cdc13\_star).$

$inh\_th(cdc2Cdc13_star, -1).$
$act\_th(cdc2Cdc13, -1).$

$inhibits(sk, ste9, T).$
$inhibits(sk, rum1, T).$
$inhibits(cdc2Cdc13, ste9, T).$
$inhibits(ste9, cdc2Cdc13, T).$
$inhibits(cdc2Cdc13, rum1, T).$
$inhibits(rum1, cdc2Cdc13, T).$
$inhibits(pp, cdc25, T).$
$inhibits(cdc2Cdc13, wee1Mik1, T).$
$inhibits(slp1, cdc2Cdc13, T).$
$inhibits(ste9, cdc2Cdc13\_star, T).$
$inhibits(cdc2Cdc13\_star, ste9, T).$
$inhibits(rum1, cdc2Cdc13\_star, T).$
$inhibits(cdc2Cdc13\_star, rum1, T).$
$inhibits(slp1, cdc2Cdc13\_star, T).$
$inhibits(wee1Mik1, cdc2Cdc13\_star, T).$

$activates(cdc2Cdc13, cdc25, T).$
$activates(pp, ste9, T).$
$activates(pp, rum1, T).$
$activates(pp, wee1Mik1, T).$
$activates(slp1, pp, T).$
$activates(cdc25, cdc2Cdc13\_star, T).$
$activates(cdc2Cdc13\_star, slp1, T).$

2 Appendix: counting the number of activating and inhibiting links

To count the number of activating and inhibiting links we follow the standard technique of aggregation, explained in [1]. Let us consider the case when we counting *activates*/3 predicate, i.e. the activation links. The idea of the approach is that first, we should assign a unique id to each of the predicates, that should be counted together, and then to iterate through these ids to perform the actual calculation.

First step is to assign indices to the activation predicates.

$$assigned\_act(activates(Y, X, T), J) \leftarrow activates(Y, X, T), \\ not\ -assigned\_act(activates(Y, X, T), J), act(Y, T).$$

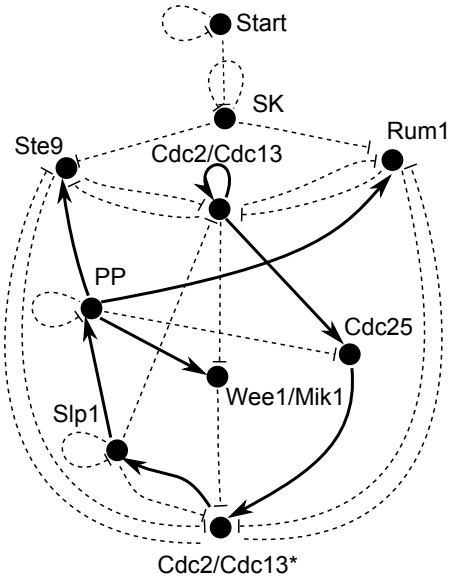

Figure 1: The dynamical network model of the *Fission Yeast* cell cycle. Taken from [2].

Check that two different indices are not assigned to the same entity and that two different entities are not assigned the same index.

$$
\begin{aligned}
-assigned\_act(activates(Y,X,T),J) \;&\leftarrow\; assigned\_act(activates(Y,X,T),J1),\\
&\quad\; J! = J1, activates(Y,X,T), act(Y,T).\\
-assigned\_act(activates(Z,X,T),J) \;&\leftarrow\; assigned\_act(activates(Y,X,T),J),\\
&\quad\; Z! = Y, activates(Z,X,T), prot(Y), act(Z,T).
\end{aligned}
$$

Now we are sure that every predicate is assigned unique index. Then we make sure that there is no gap in the index assignment so that we can iterate on it further, and that indices are starting from 1.

$$
\begin{aligned}
numbered\_act(X,J,T) \;&\leftarrow\; assigned\_act(activates(Y,X,T),J),\\
&\quad\; activates(Y,X,T), act(Y,T).\\
&\leftarrow\; numbered\_act(X,J+1,T),\\
&\quad\; not\ numbered\_act(X,J,T),\\
&\quad\; J >= 1, prot(X).\\
one\_is\_assigned\_act(X,T) \;&\leftarrow\; assigned\_act(activates(Y,X,T),1),\\
&\quad\; activates(Y,X,T,act(Y,T).\\
&\leftarrow\; activates(Y,X,T), not\ one\_is\_assigned\_act(X,T), act(Y,T).
\end{aligned}
$$

Now we can describe a recursive procedure of link counting and the stopping condition of this procedure.

$$
\begin{aligned}
last\_activation\_index(X,J,T) \;&\leftarrow\; activates(Y,X,T),\\
&\quad\; numbered\_act(X,J,T),\\
&\quad\; not\ numbered\_act(X,J+1,T), act(Y,T).\\
count\_activation\_links(X,1,T) \;&\leftarrow\; assigned\_act(activates(Y,X,T),1),\\
&\quad\; activates(Y,X,T), act(Y,T).\\
count\_activation\_links(X,J+1,T) \;&\leftarrow\; J>0, count\_activation\_links(X,J,T),\\
&\quad\; assigned\_act(activates(Y,X,T),J+1),\\
&\quad\; activates(Y,X,T), act(Y,T).\\
\#\_act(X,J,T) \;&\leftarrow\; count\_activation\_links(X,J,T),\\
&\quad\; last\_activation\_index(X,J,T), prot(X).
\end{aligned}
$$

Finally, we write a special case rule, that is fired when the protein have no activation links.

$$\begin{array}{lcl} \#\_act(X, 0, T) & \leftarrow & not\ has\_activation\_links(X, T), prot(X). \\ has\_activation\_links(X, T) & \leftarrow & activates(Y, X, T), act(Y, T). \end{array}$$

The same procedure applies to count the number of inhibition links.

References

[1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[2] M. I. Davidich and S. Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS ONE*, 3(2), 2008.