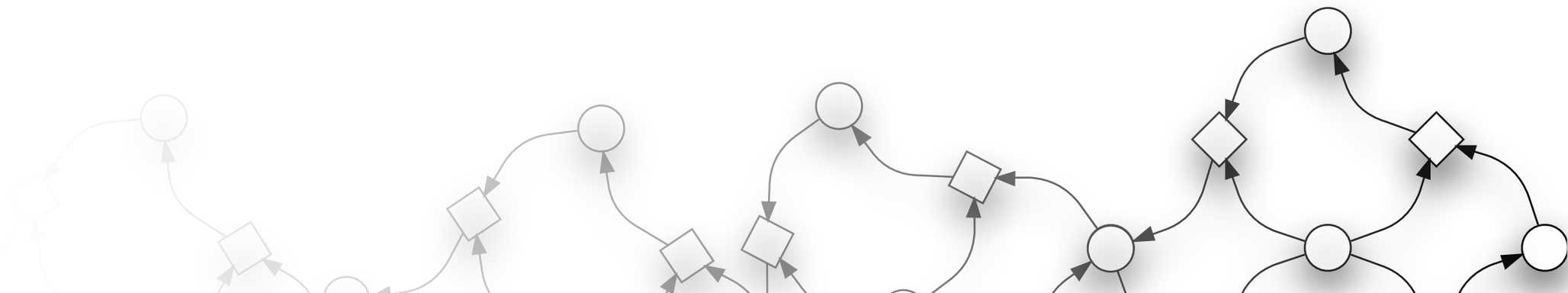
BioPPN 2015, Brussel/Belgium - 22nd June 2015

Spatial Modelling based on the BMK-Framework

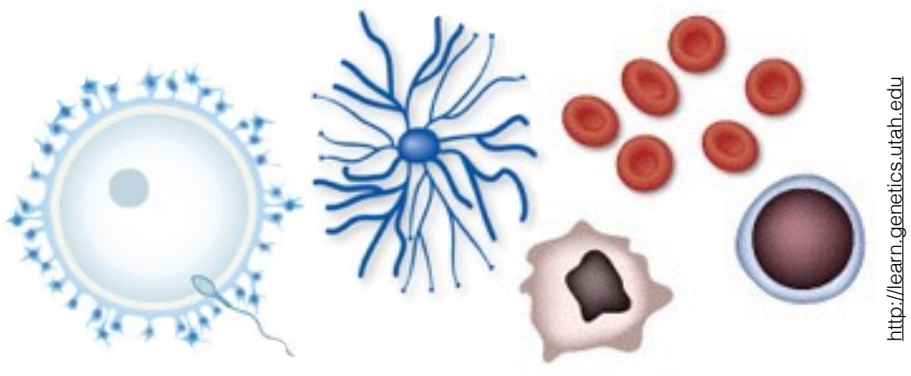
Mary-Ann Blätke[1], Christian Rohr[2]

[1] Otto-von-Guericke-University Magdeburg, Germany
[2] Brandenburg University of Technology Cottbus, Germany



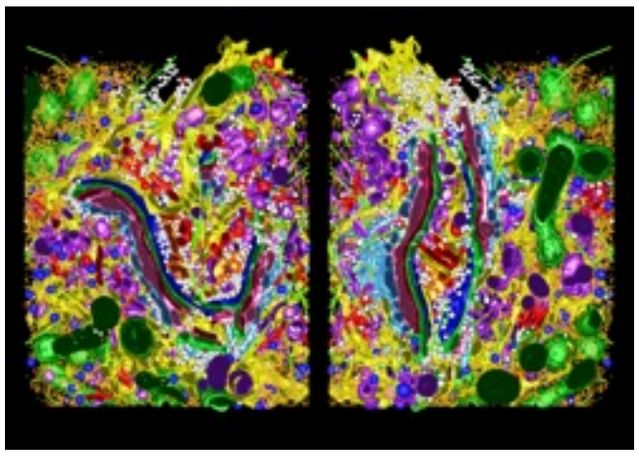
Background

- Systems Biology encouraged the modelling of bio-systems
  - detailed mechanistic & kinetic descriptions
  - ignore spatial aspects
- But space matters:



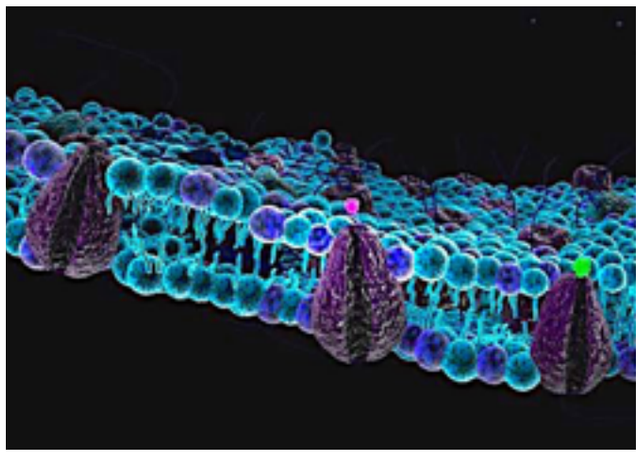
http://learn.genetics.utah.edu

cell size & shape



http://learn.genetics.utah.edu

compartmentalisation,
pools & organisation



http://learn.genetics.utah.edu

membrane composition

- Integration of spatial aspects in existing models is not straightforward
  - ➡ models have to be rebuild

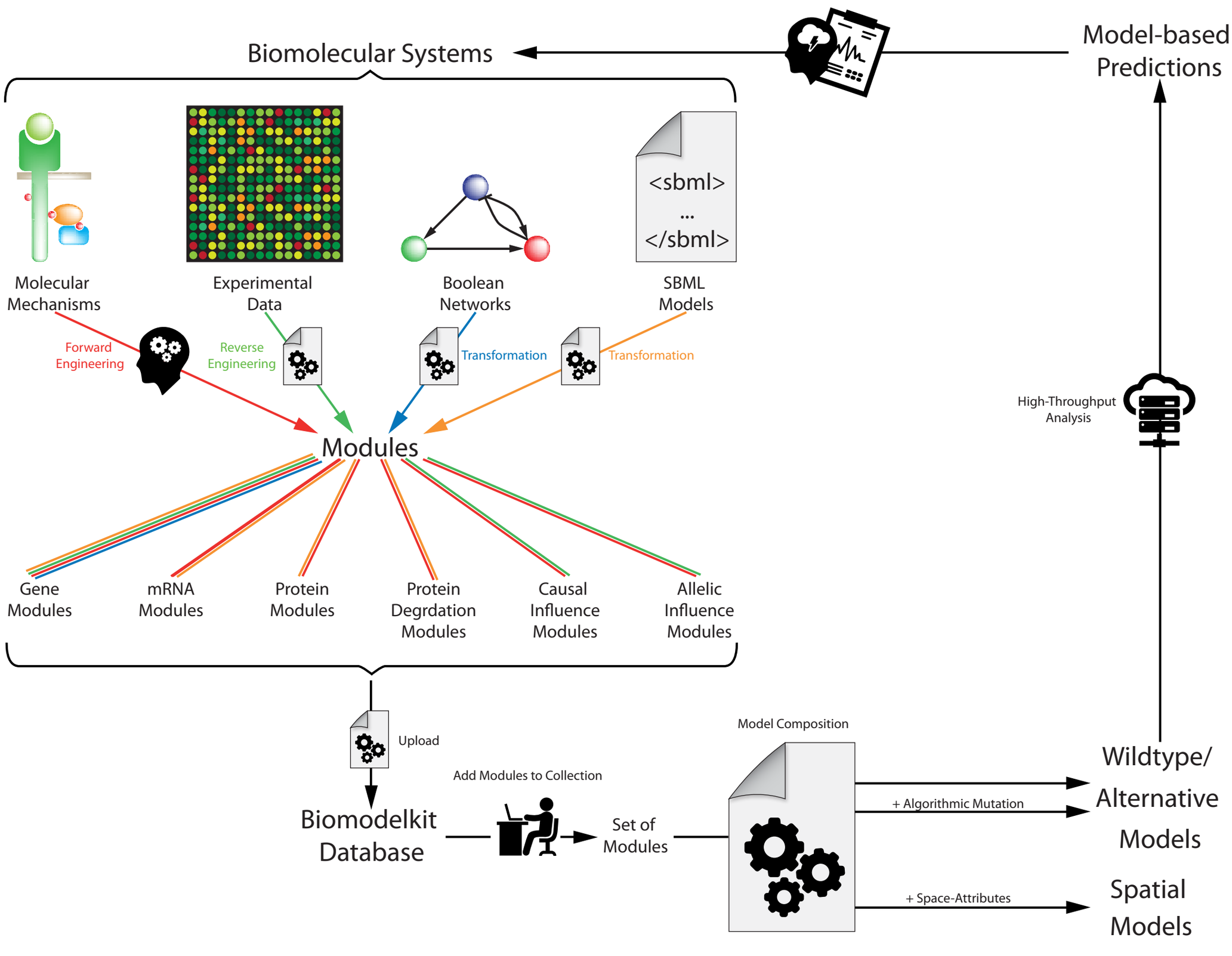
Our Idea

- Extend models with spatial information to keep track of position and movement
- Biomolecule-centred modularisation of models

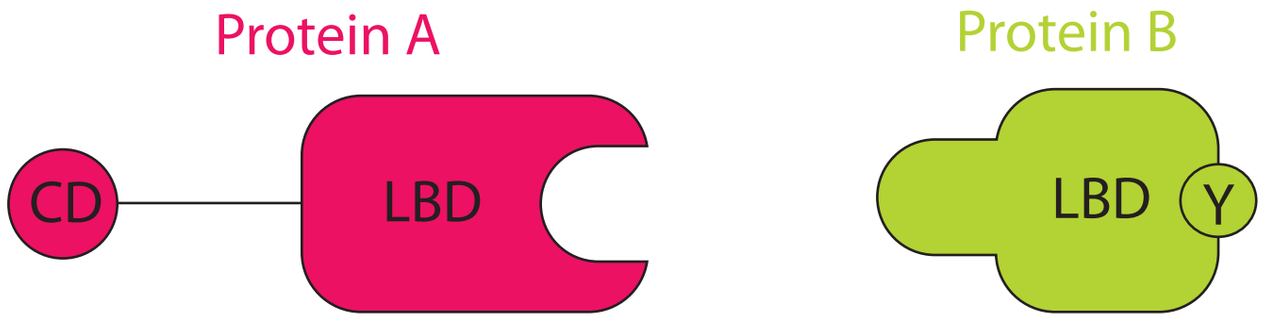
➡Biomodelkit Framework

A module describes the functionality and interaction of a biomolecule in the form of a Petri net

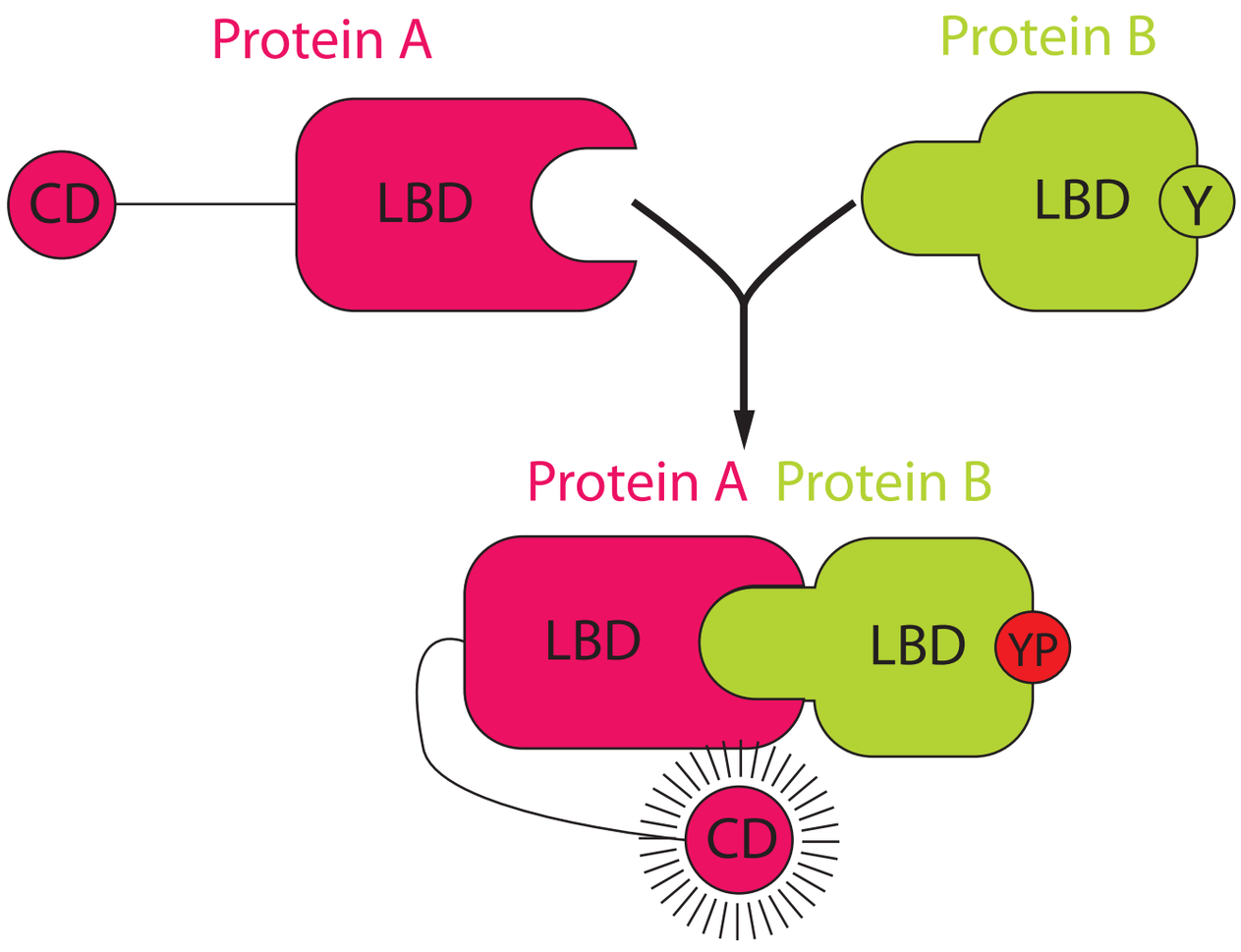
BMK-Framework



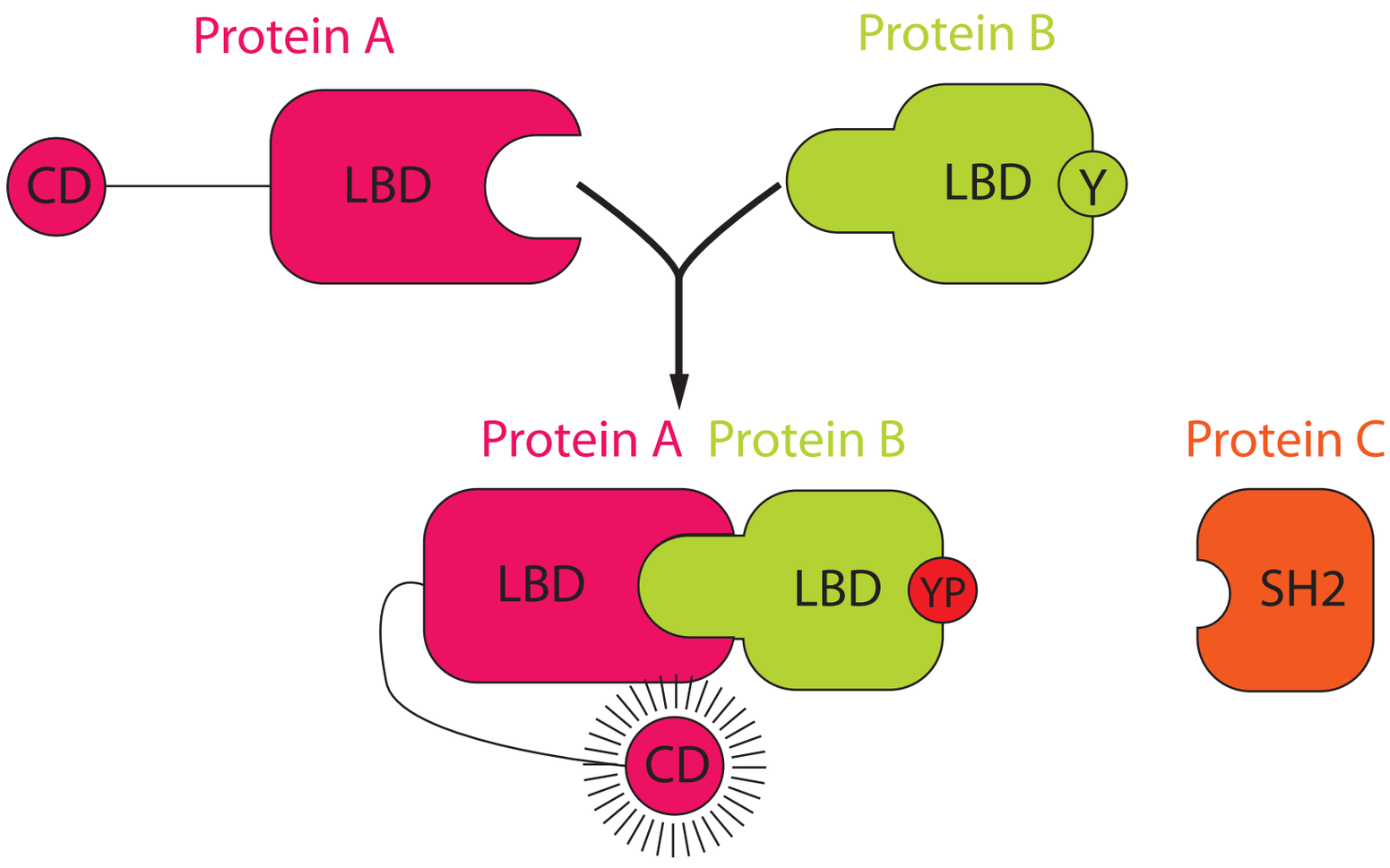
Running Example



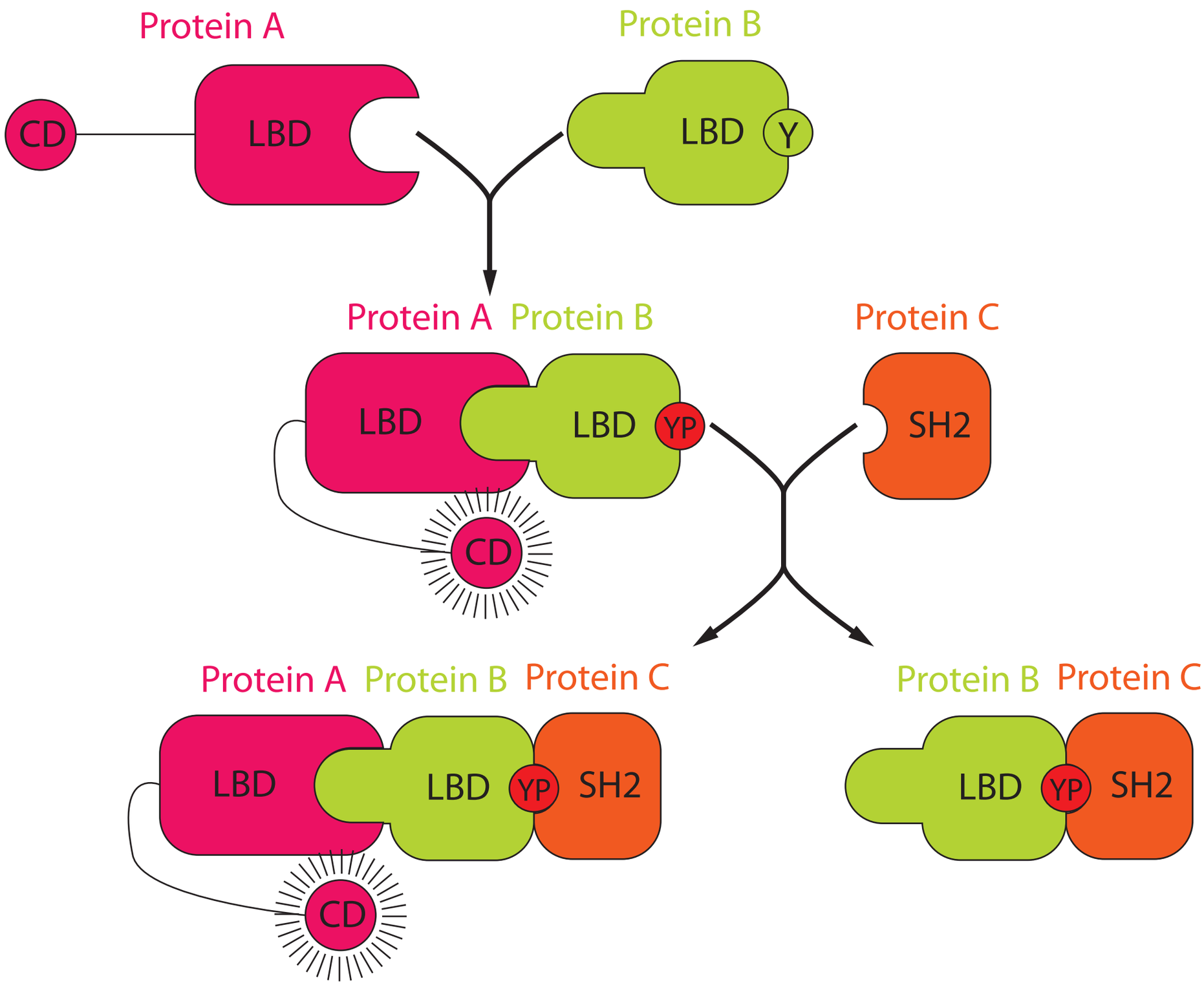
Running Example



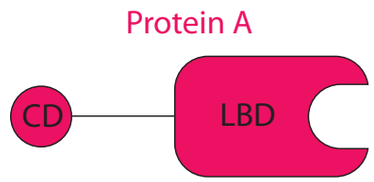
Running Example



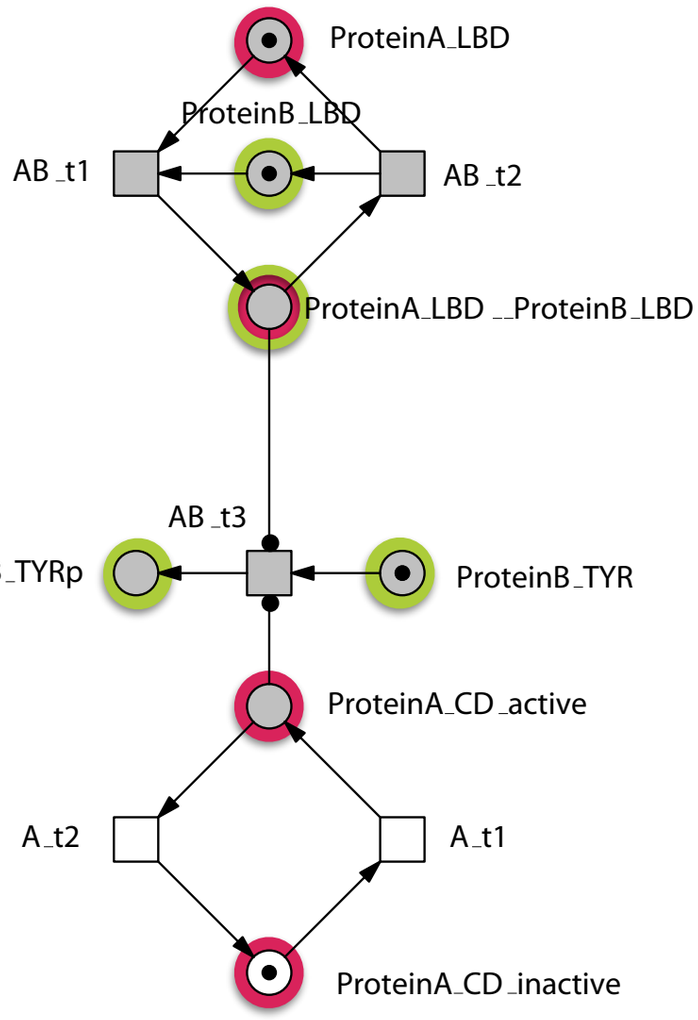
Running Example

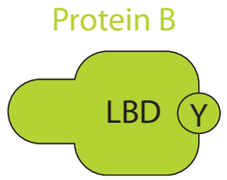


Modules of the Running Example

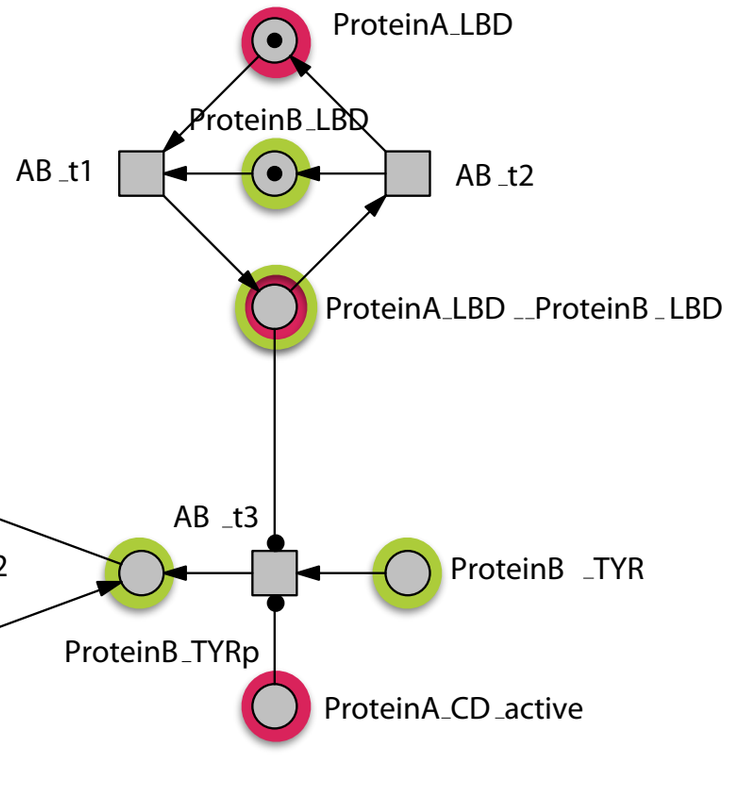


Module of Protein A



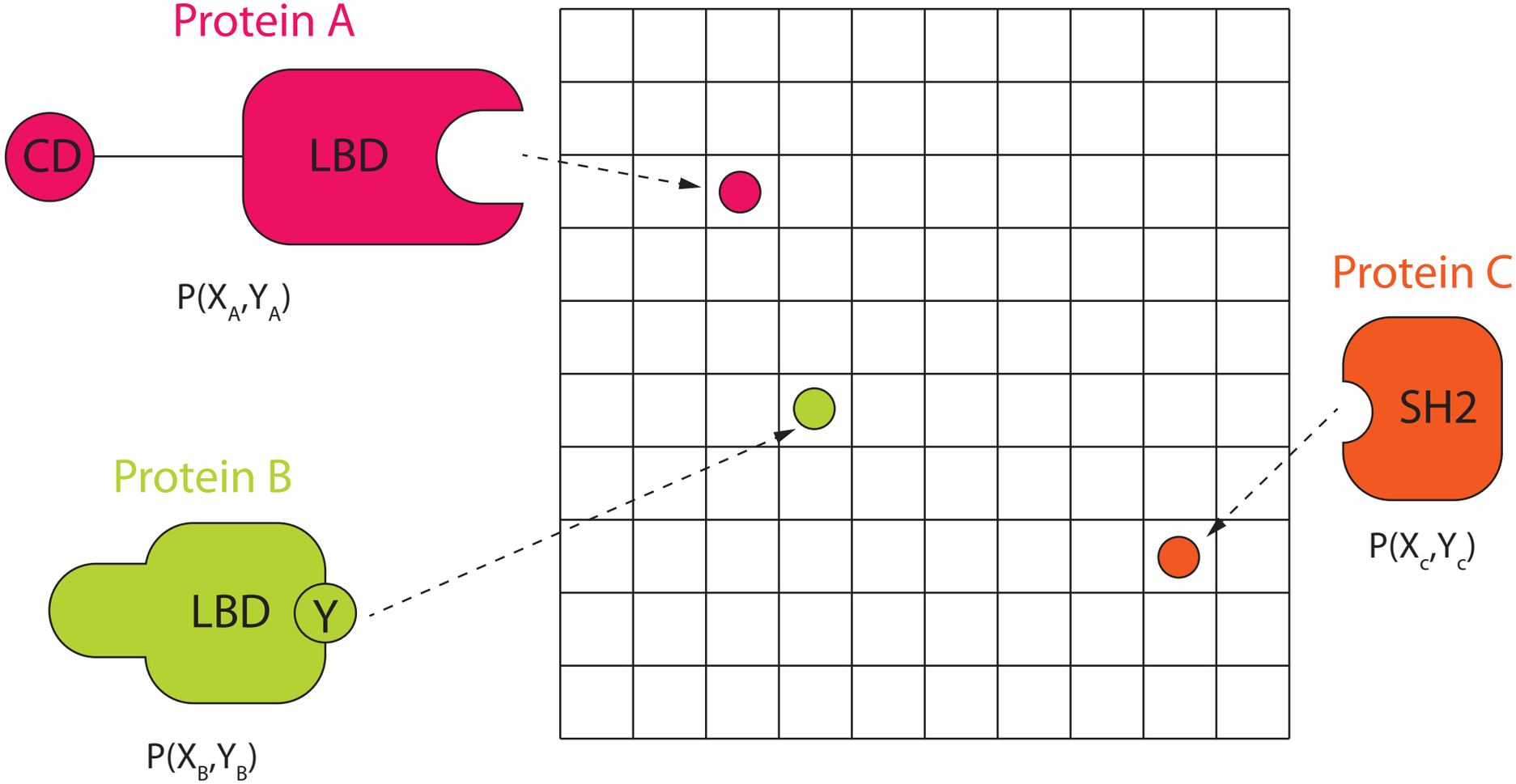


Module of Protein B



Step 1: Explicit Encoding of Local Positions

- Define a grid for each component (1D,2D or 3D)
  - e.g. 2D Grid: xDimA = 10, yDimA = 10
- Set a local position for each component
  - e.g. A(3,8), B(4,5), C(9,3)



Step 1: Explicit Encoding of Local Positions

- Define a grid for each component (1D,2D or 3D)
  - e.g. 2D Grid: xDimA = 10, yDimA = 10
- Set a local position for each component
  - e.g. A(3,8), B(4,5), C(9,3)

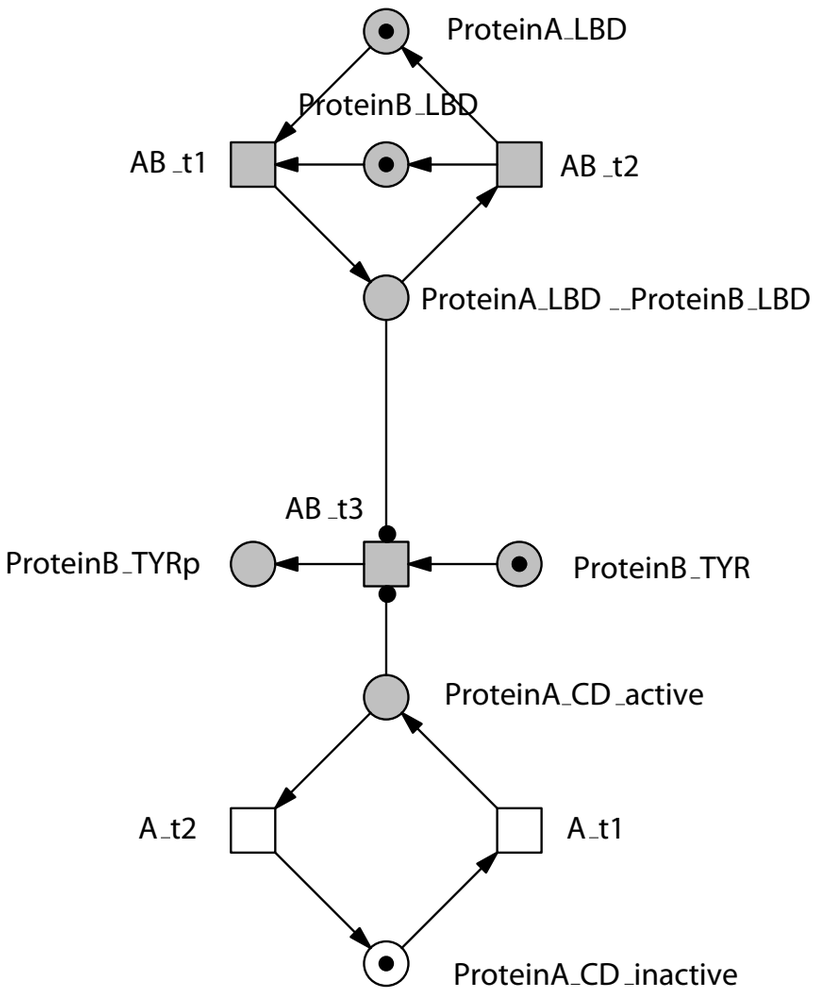
Step 1: Explicit Encoding of Local Positions

- Define a grid for each component (1D,2D or 3D)
  - e.g. 2D Grid: xDimA = 10, yDimA = 10
- Set a local position for each component
  - e.g. A(3,8), B(4,5), C(9,3)

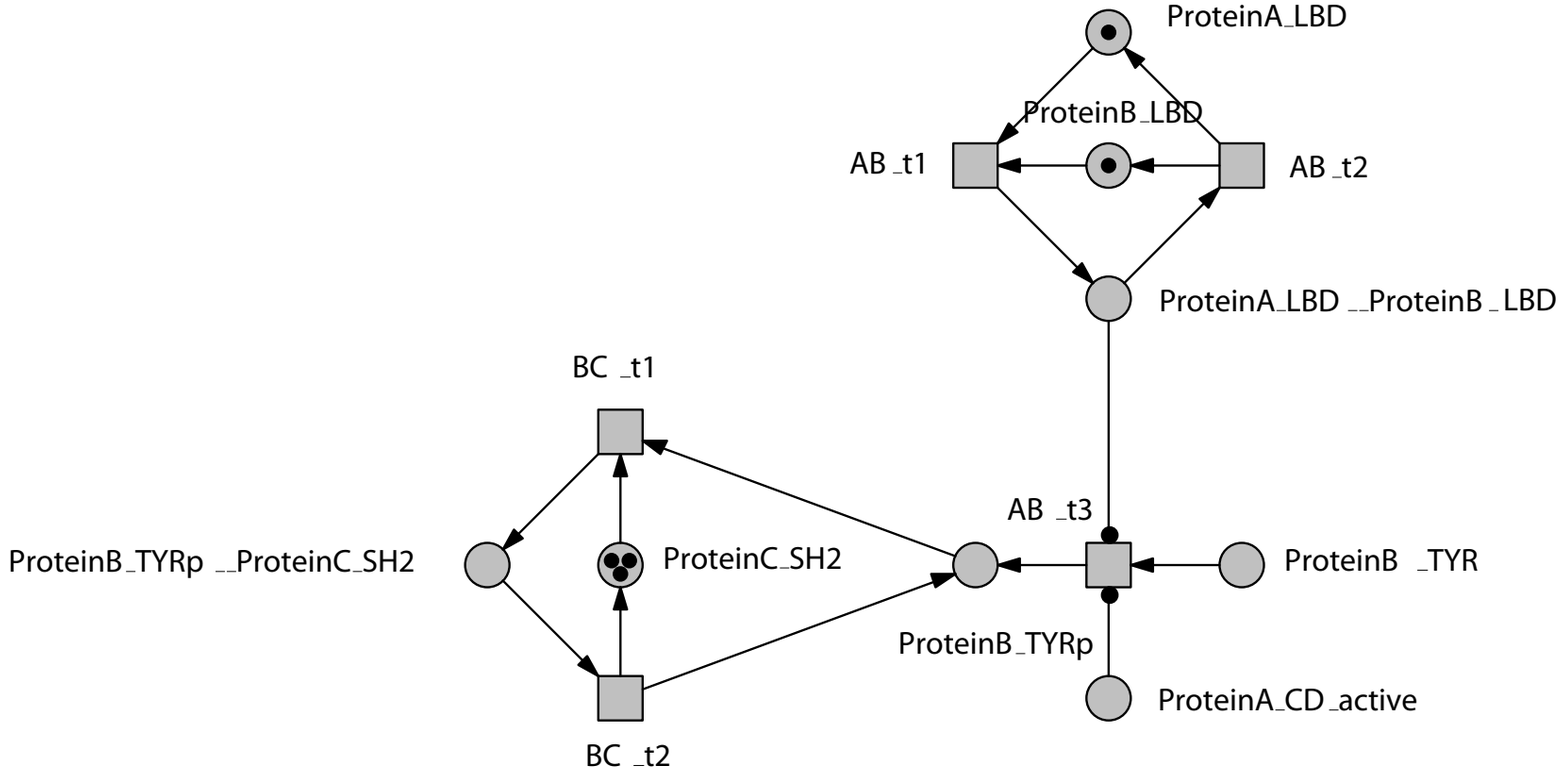
Add a place for each component and axis

Step 1: Explicit Encoding of Local Positions

Module of Protein A



Module of Protein B



XY-Position of Protein A

ProteinA_X ProteinA_Y

XY-Position of Protein B

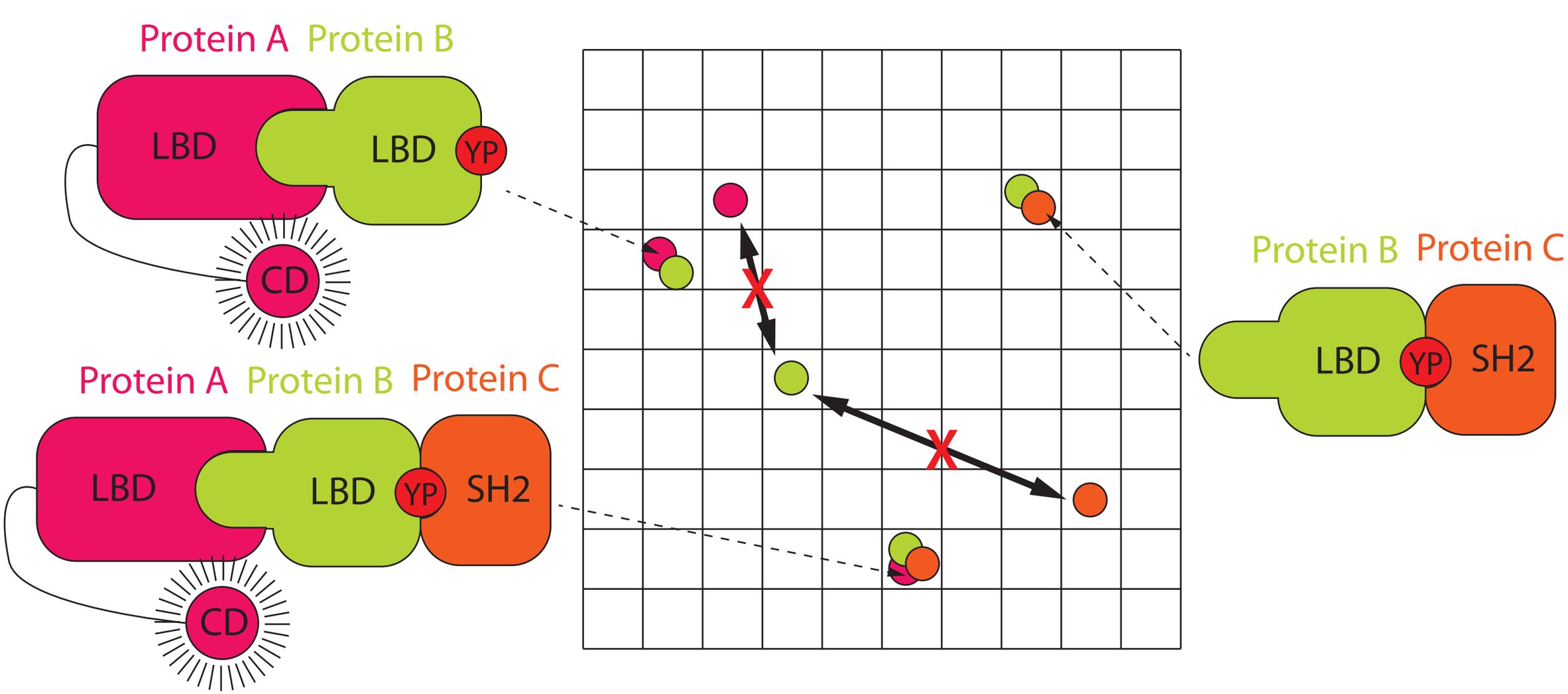
ProteinB_X ProteinB_Y

XY-Position of Protein C

ProteinC_X ProteinC_Y

Step 2: Local Restriction of Interactions

- Components can only interact if they fulfil a defined neighbourhood relation
- Define a neighbourhood relation
  - e.g. local positions of components must be identical



Step 2: Local Restriction of Interactions

- Components can only interact if they fulfil a defined neighbourhood relation
- Define a neighbourhood relation
  - e.g. local positions of components must be identical

Step 2: Local Restriction of Interactions

- Components can only interact if they fulfil a defined neighbourhood relation
- Define a neighbourhood relation
  - e.g. local positions of components must be identical

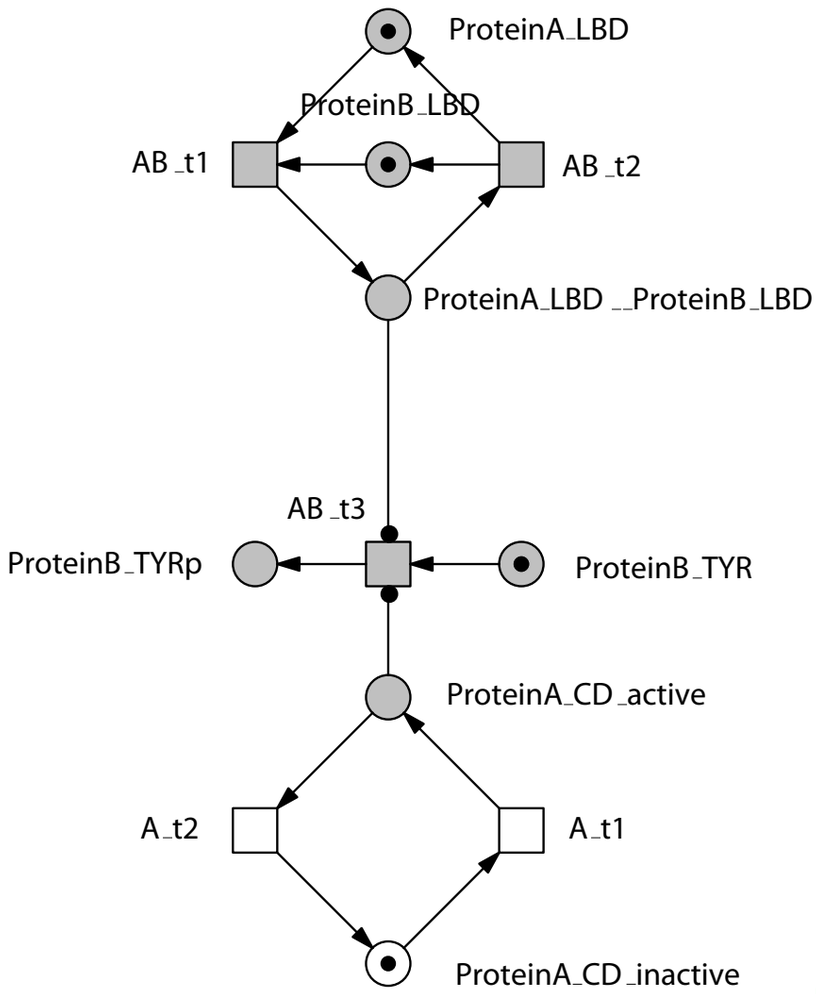
For each transition representing an interaction:

- add coordinate places of interacting components

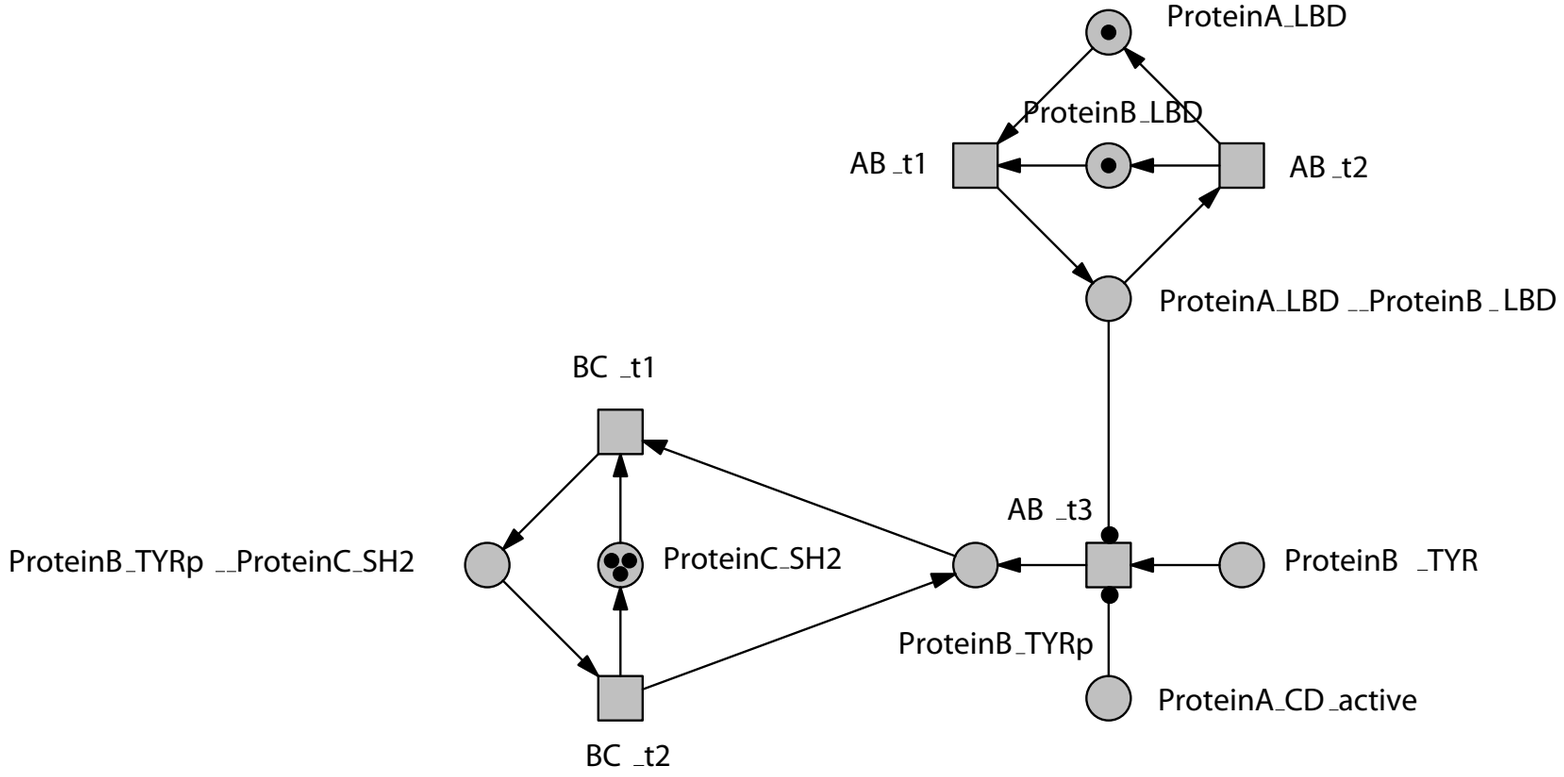
- multiply firing-rate with a boolean expression evaluating the distance between the interacting components

Step 2: Local Restriction of Interactions

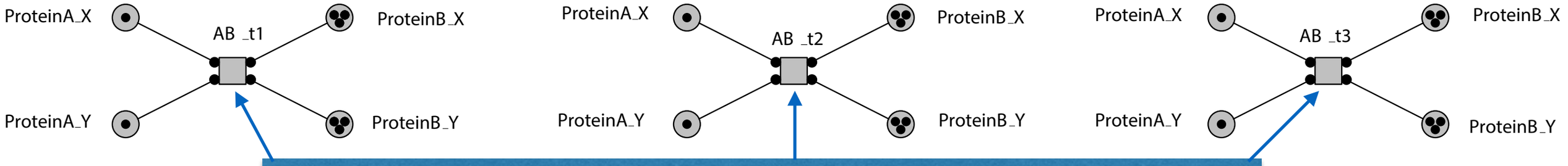
Module of Protein A



Module of Protein B



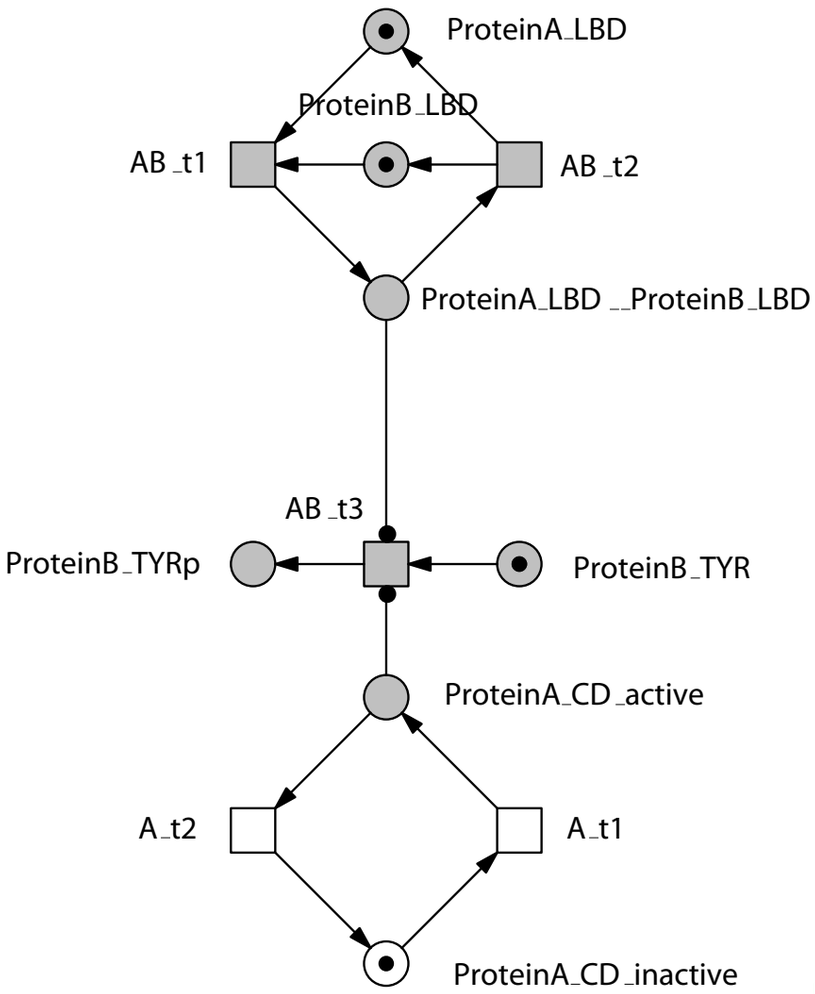
Restricted Interaction - Only if ProteinA (X,Y) = ProteinB (X,Y)



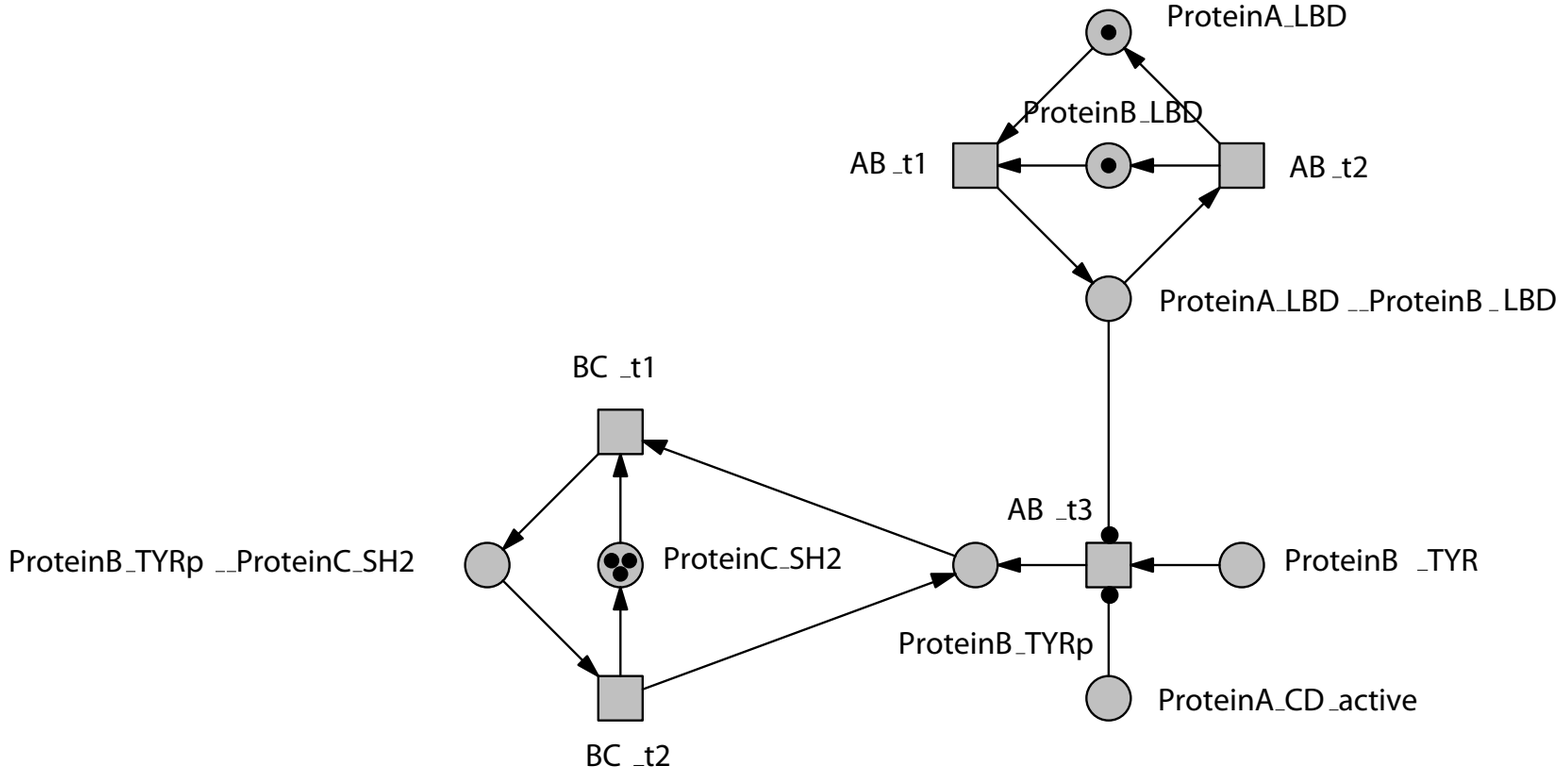
(ProteinA_X = ProteinB_X) & (ProteinA_Y = ProteinB_Y)*h(t)

Step 2: Local Restriction of Interactions

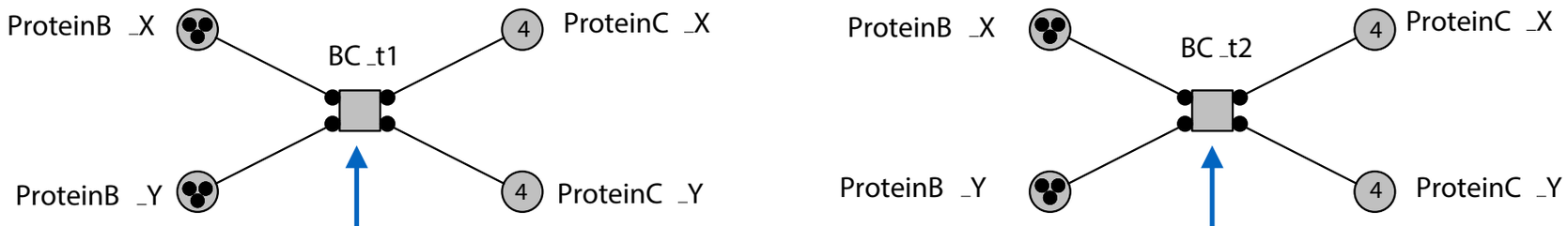
Module of Protein A



Module of Protein B



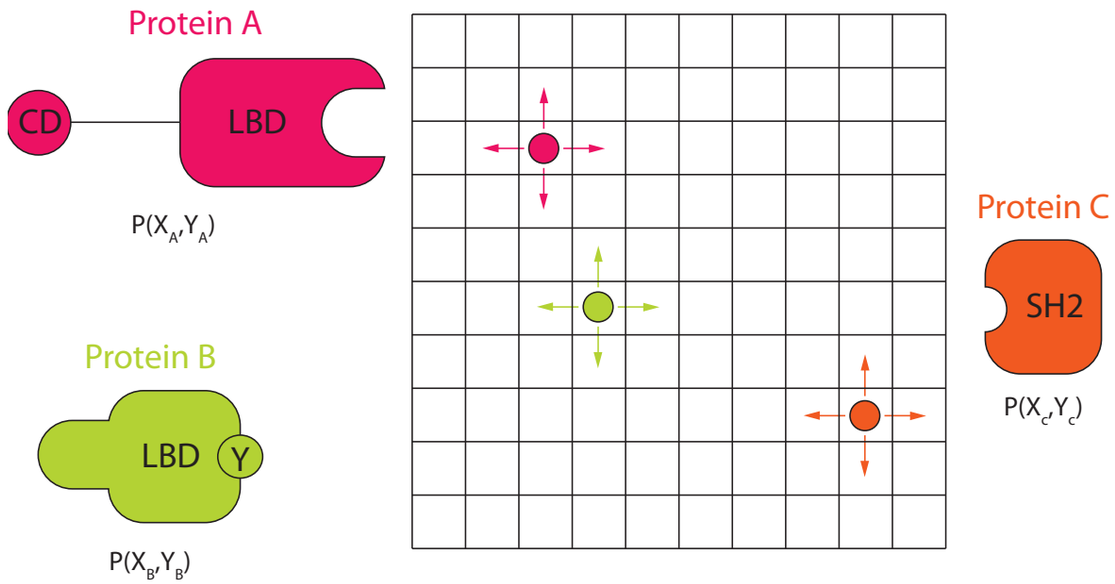
Restricted Interaction - Only if ProteinB (X,Y) = ProteinC (X,Y)



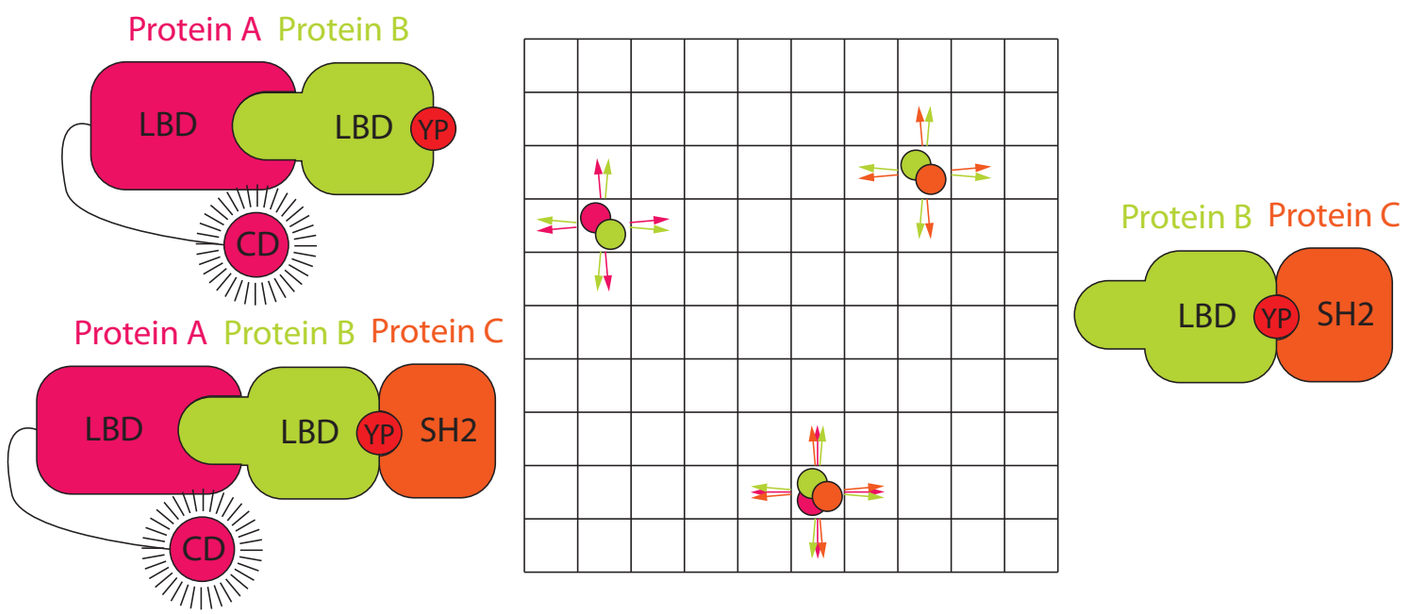
(ProteinB_X = ProteinC_X) & (ProteinB_Y = ProteinC_Y)*h(t)

Step 3: Explicit Encoding of Local Position Changes

Case 1: Movement of components as a single entity

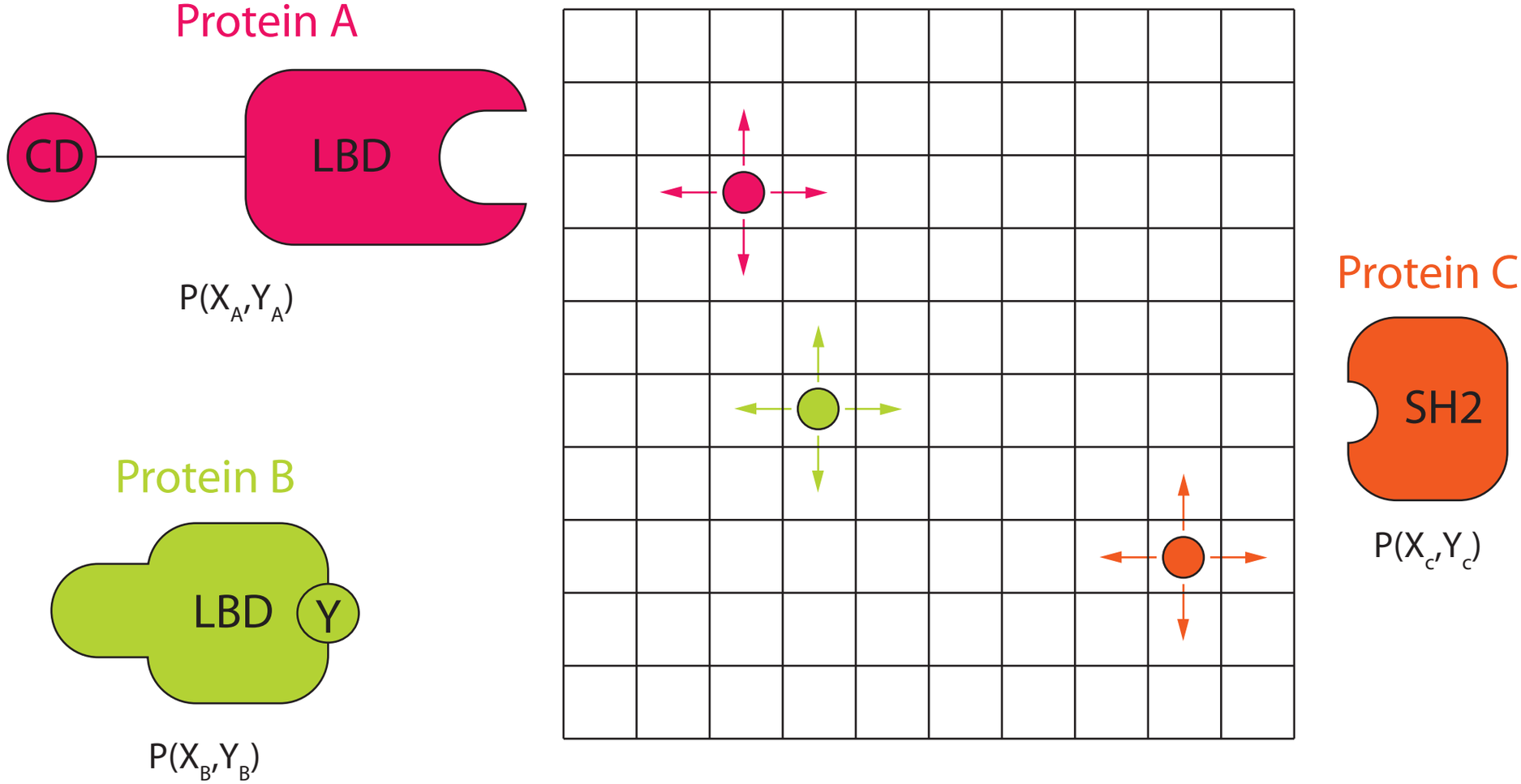


Case 2: Movement of components as complex



Case 1: Movement of components as a single entity

- Movement along the axes in respect to the defined grid size
  - e.g. 2D-Grid -> 2 direction of movement per axis
- Movement only if all interaction sites are unused
  - e.g. Protein B is not allowed to interact with Protein A or Protein C



Case 1: Movement of components as a single entity

- Movement along the axes in respect to the defined grid size
  - e.g. 2D-Grid -> 2 direction of movement per axis
- Movement only if all interaction sites are unused
  - e.g. Protein B is not allowed to interact with Protein A or Protein C

Case 1: Movement of components as a single entity

- Movement along the axes in respect to the defined grid size
  - e.g. 2D-Grid -> 2 direction of movement per axis
- Movement only if all interaction sites are unused
  - e.g. Protein B is not allowed to interact with Protein A or Protein C

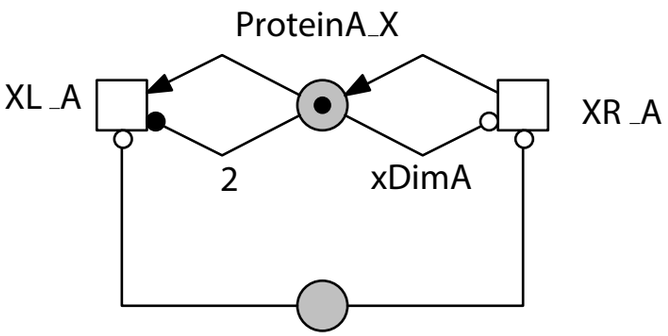
For each component:

- add two transitions per axis to increase/decrease the value of the coordinate places in respect to the grid size

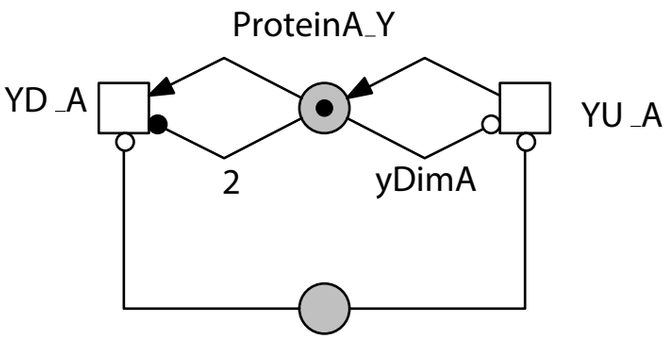
- connect places representing interaction states of the components via inhibitory arcs

Case 1: Movement of components as a single entity

Movement of Protein A

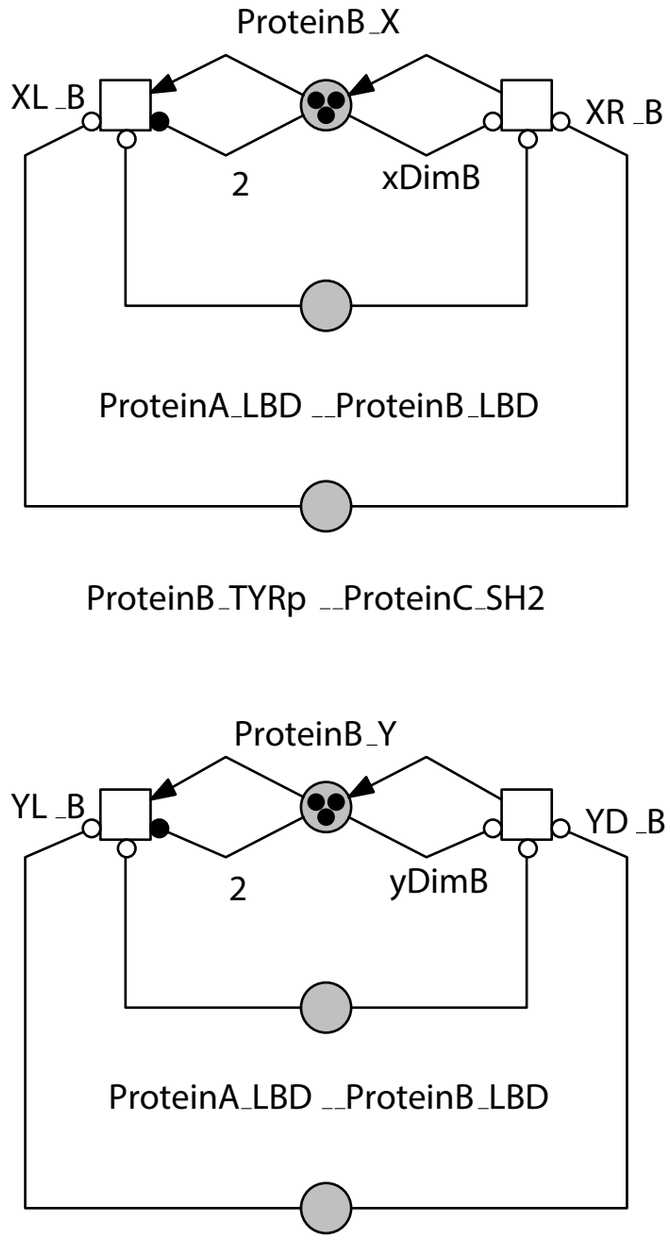


ProteinA_LBD __ProteinB_LBD



ProteinA_LBD __ProteinB_LBD

Movement of Protein B



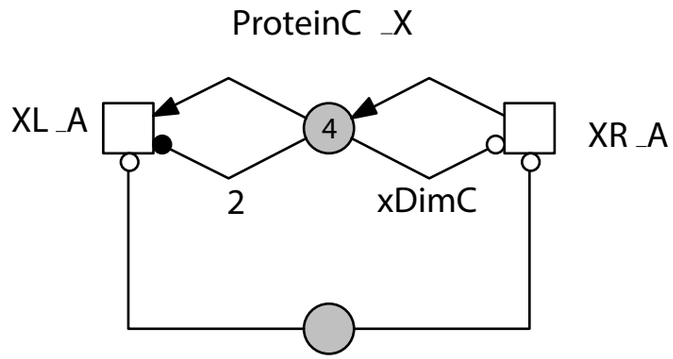
ProteinA_LBD __ProteinB_LBD

ProteinB_TYRp __ProteinC_SH2

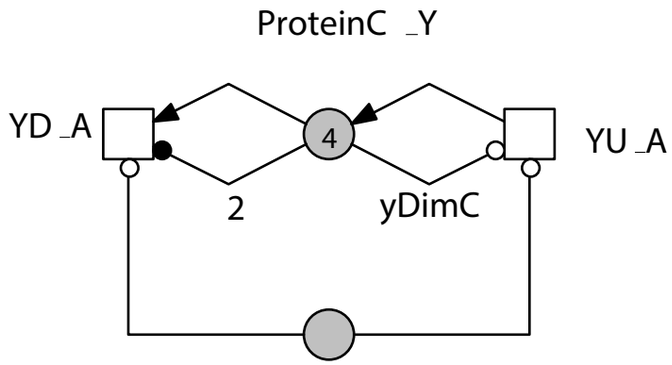
ProteinA_LBD __ProteinB_LBD

ProteinB_TYRp __ProteinC_SH2

Movement of Protein C



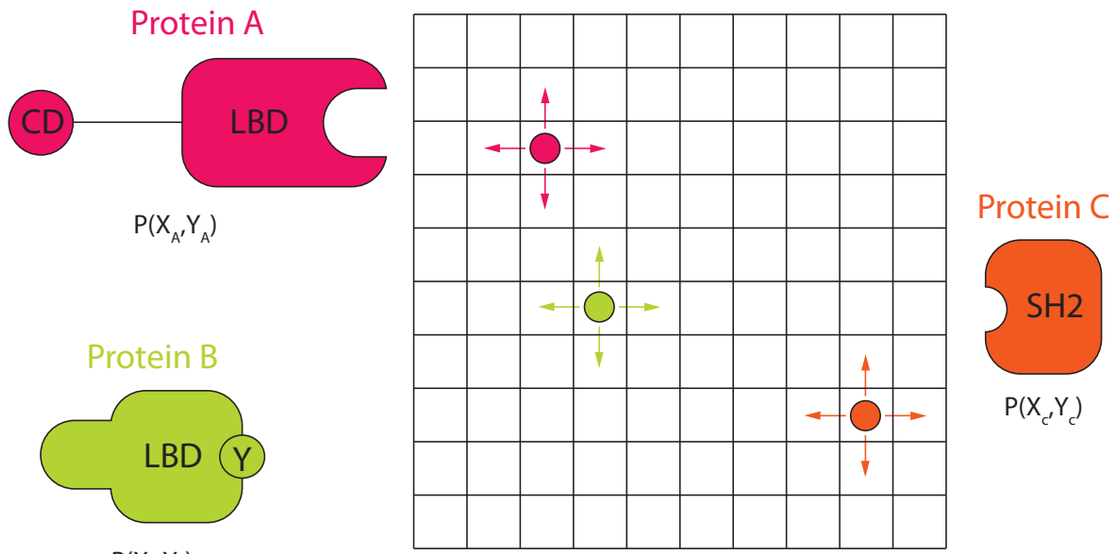
ProteinB_TYRp __ProteinC_SH2



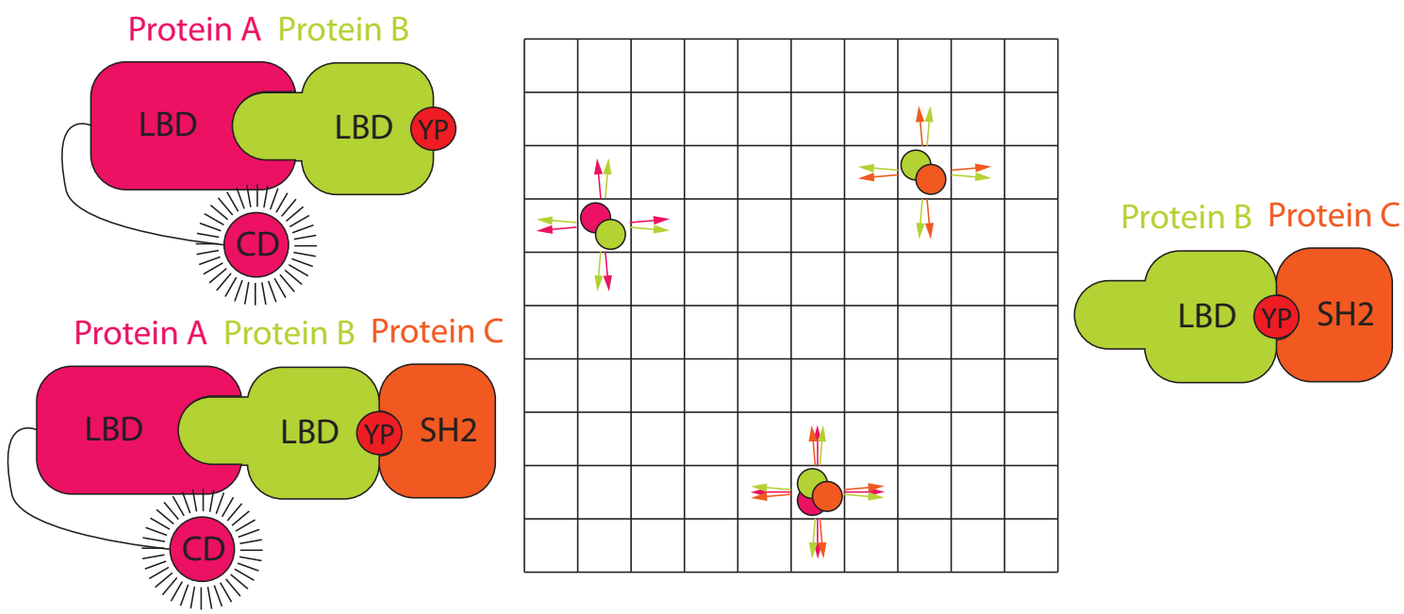
ProteinB_TYRp __ProteinC_SH2

Step 3: Explicit Encoding of Local Position Changes

Case 1: Movement of components as a single entity

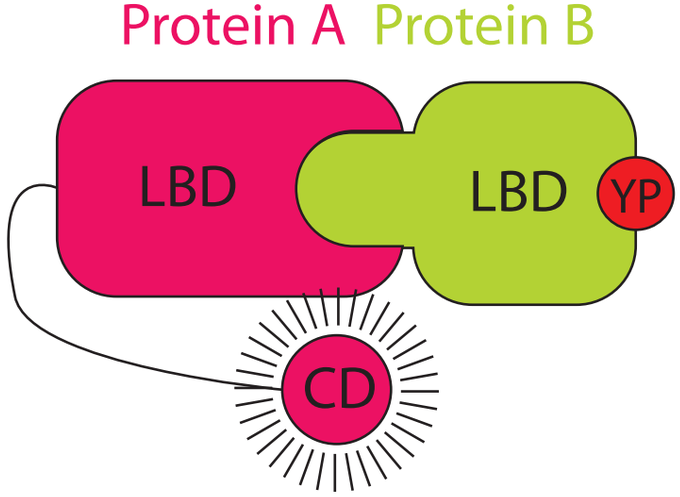


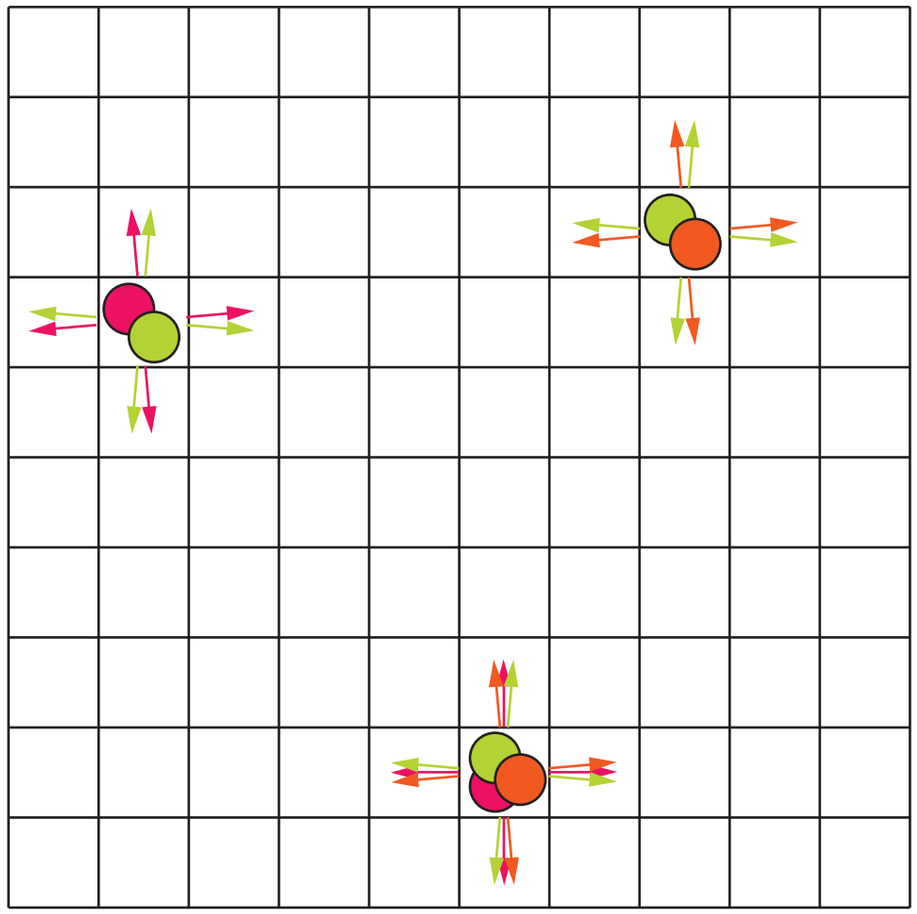
Case 2: Movement of components as complex

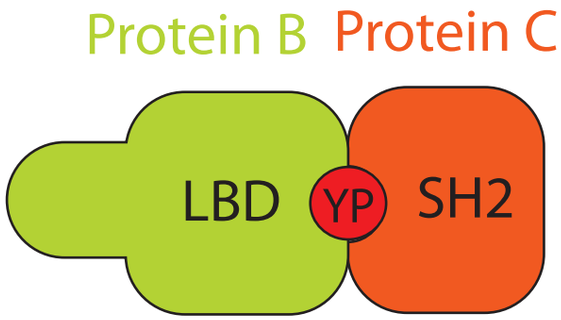


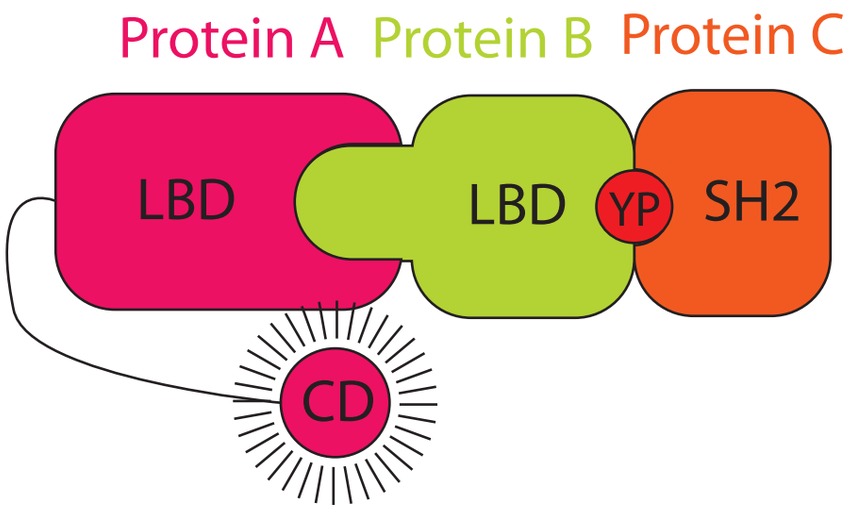
Case 2: Movement of components as complex

- Determine all possible complexes
  - e.g. Protein A + Protein B, Protein B + Protein C, Protein A + Protein B + Protein C
- Movement only if the corresponding interaction sites of interacting components in a complex are used and all other are unused
  - e.g. Protein A + Protein B can only move if they interact with each other, but not with Protein C
- Update local positions of all components forming a complex simultaneously









Case 2: Movement of components as complex

- Determine all possible complexes
  - e.g. Protein A + Protein B, Protein B + Protein C, Protein A + Protein B + Protein C
- Movement only if the corresponding interaction sites of a complex are used and all other are unused
  - e.g. Protein A + Protein B can only move if they interact with each other, but not with Protein C
- Update local positions of all components forming a complex simultaneously

Case 2: Movement of components as complex

- Determine all possible complexes
    - e.g. Protein A + Protein B, Protein B + Protein C, Protein A + Protein B + Protein C
- Movement only if the corresponding interaction sites of a complex are used and all other are unused
    - e.g. Protein A + Protein B can only move if they interact with each other, but not with Protein C
- Update local positions of all components forming a complex simultaneously

For each complex:

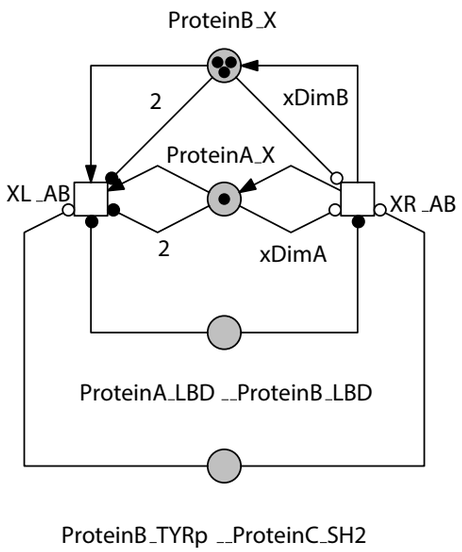
- add two transitions per axis to increase/decrease the value of the coordinate places of the interacting components in respect to the grid size

- connect places representing the respective interaction via test arcs

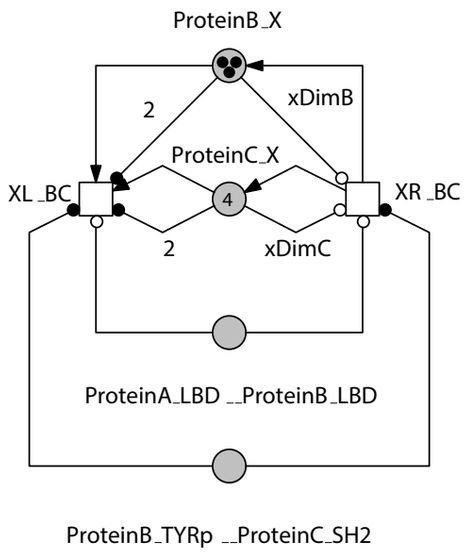
- connect places representing interaction with other components via inhibitory arcs

Case 2: Movement of components as complex

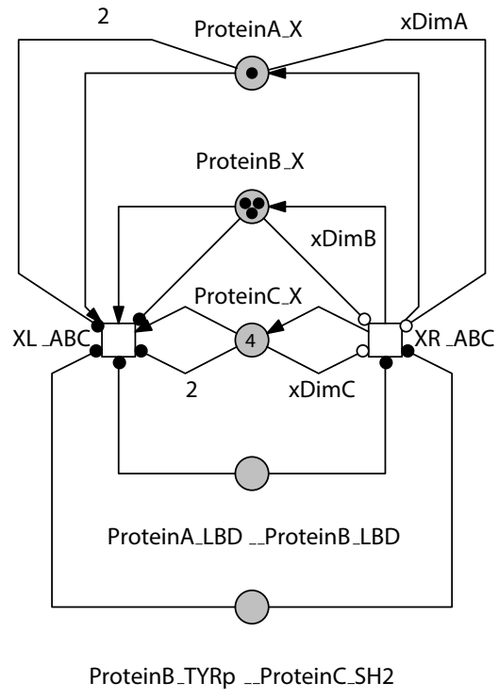
Movement of complex ProteinA _ProteinB

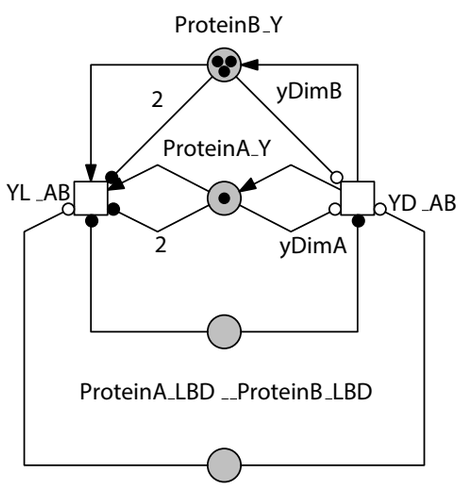


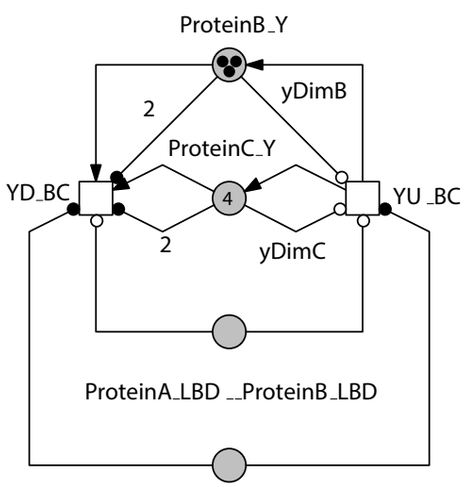
Movement of complex ProteinB _ProteinC

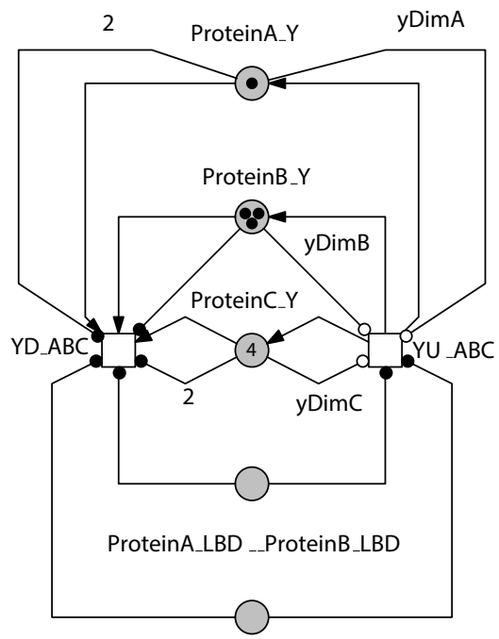


Movement of complex ProteinA _ProteinB _ ProteinC



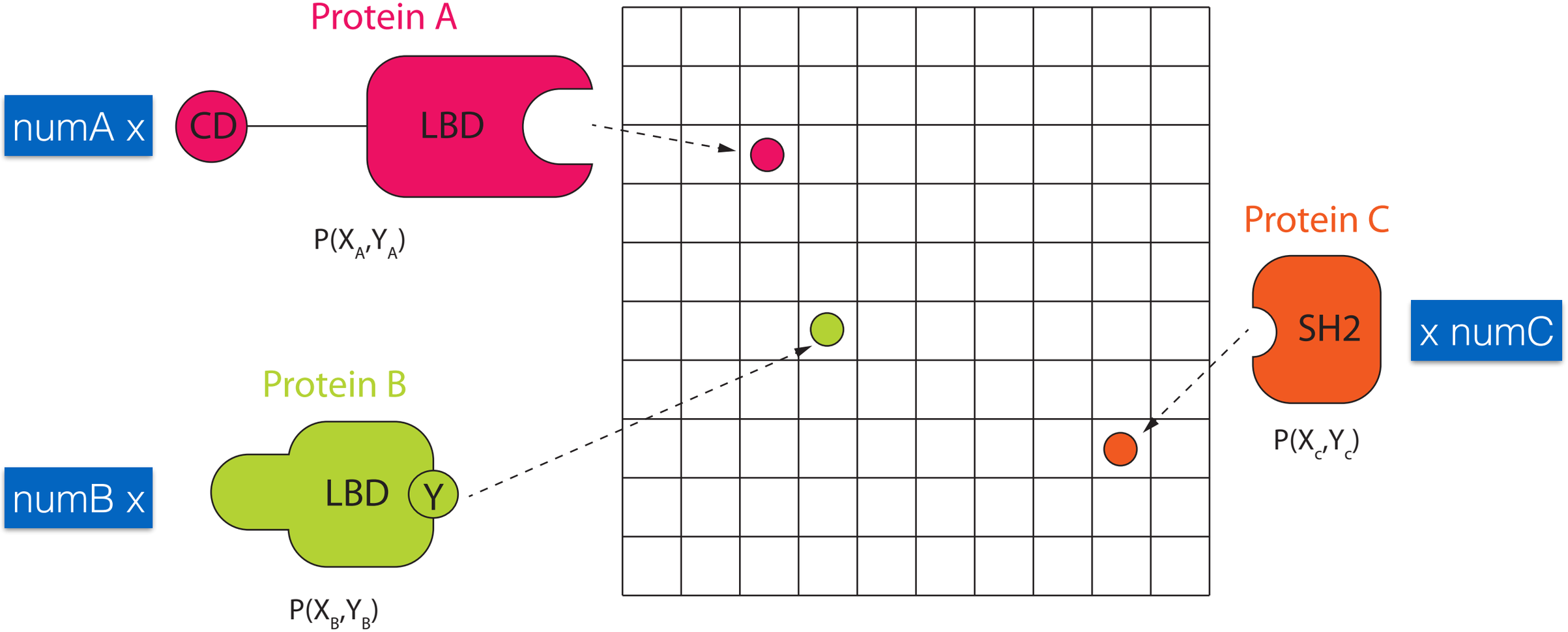






Explicit Encoding of Component Instances

- Define a number of instances for each component
  - e.g. numA = 3, numB= 3, numC = 3
- Duplicate the previously defined networks according to the number of instances



Explicit Encoding of Component Instances

- Define a number of instances for each component
  - e.g. numA = 3, numB= 3, numC = 3
- Duplicate the previously defined networks according to the number of instances

Explicit Encoding of Component Instances

- Define a number of instances for each component
  - e.g. numA = 3, numB= 3, numC = 3
- Duplicate the previously defined networks according to the number of instances

Application of coloured Petri nets:

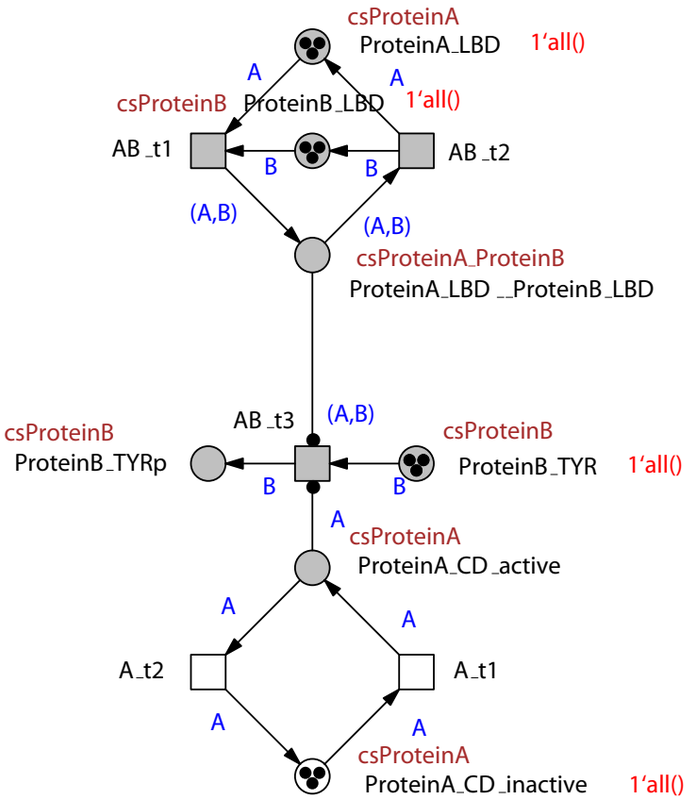
- For each component define a simple colour-set and variable,
- Add simple colour-sets to places representing non-interaction states

- For each interaction define a product colour-set based on the respective simple colour-sets
- Add product colour-sets to all places representing the respective interaction

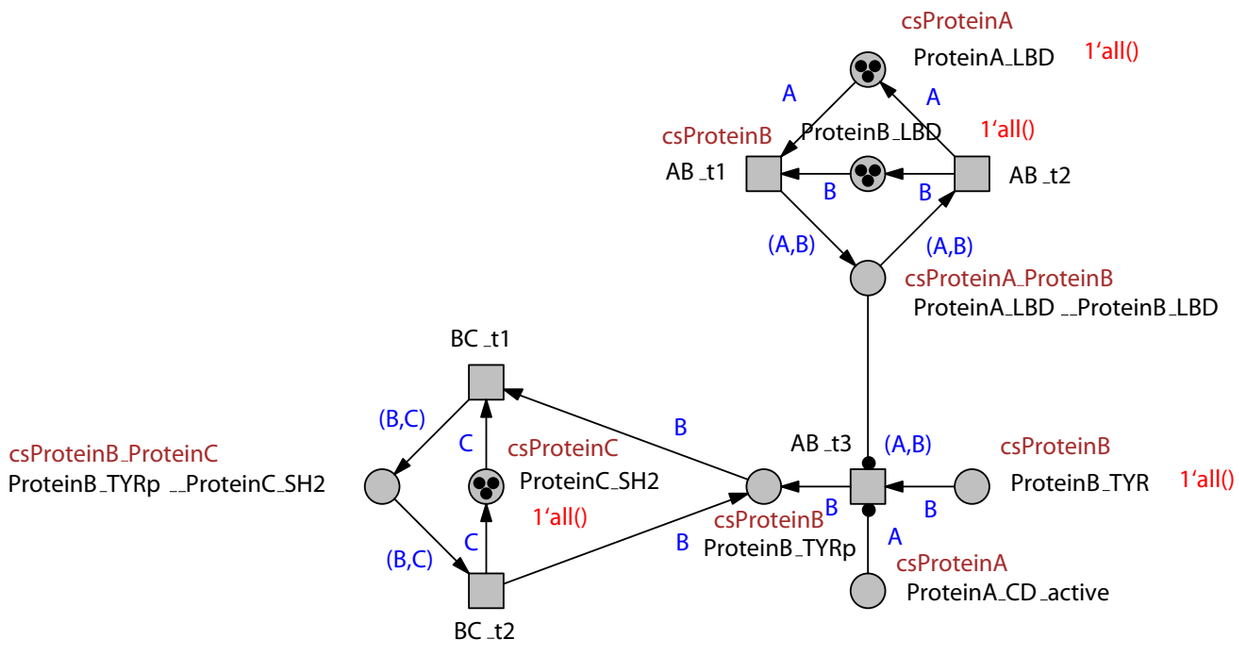
Applying Coloured Petri Nets

- Define a simple colour-set and a variable for each component
  - **Protein A** -> csProteinA:= int, 1-numA; csProteinA A
  - **Protein B** -> csProteinB:= int, 1-numB; csProteinB B
  - **Protein C** -> csProteinC:= int, 1-numC; csProteinC C

Module of Protein A



Module of Protein B



XY-Position of Protein A

csProteinA
ProteinA_X
1'all()

csProteinA
ProteinA_Y
1'all()

XY-Position of Protein B

9

csProteinB
ProteinB_X
3'all()

9

csProteinB
ProteinB_Y
3'all()

XY-Position of Protein C

12

csProteinC
ProteinC_X
4'all()

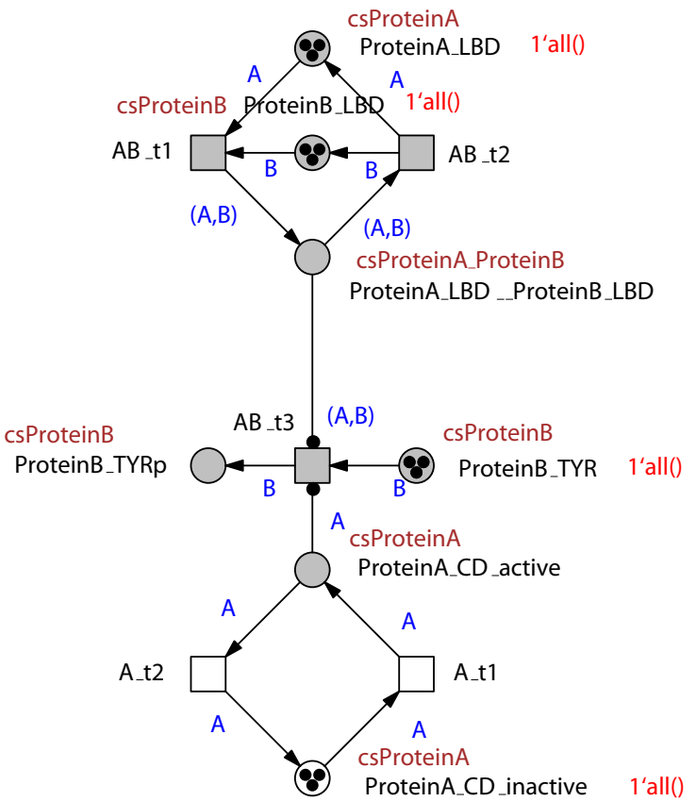
12

csProteinC
ProteinC_Y
4'all()

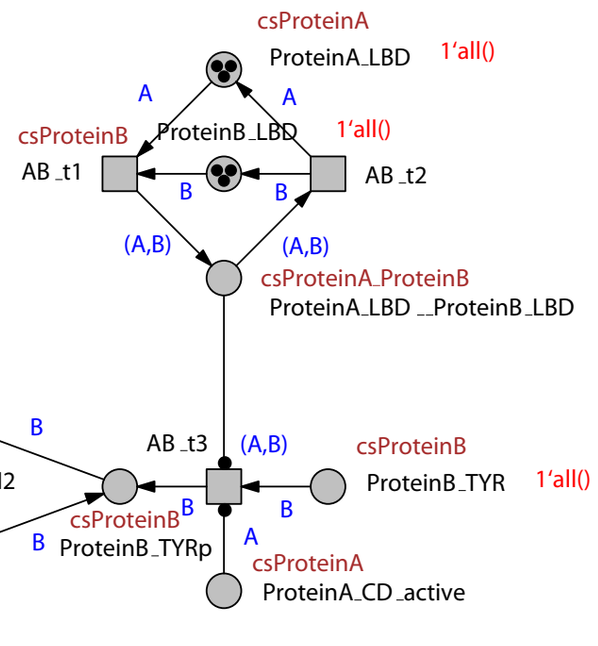
Applying Coloured Petri Nets

- Define a product colour-set for each interaction
  - **Protein A** + **Protein B** -> csProteinA__ProteinB := product csProteinA, csProteinB
  - **Protein B** + **Protein C** -> csProteinB__ProteinC := product csProteinB, csProteinC

Module of Protein A



Module of Protein B



XY-Position of Protein A

csProteinA ProteinA_X 1'all()

csProteinA ProteinA_Y 1'all()

XY-Position of Protein B

9 csProteinB ProteinB_X 3'all()

9 csProteinB ProteinB_Y 3'all()

XY-Position of Protein C

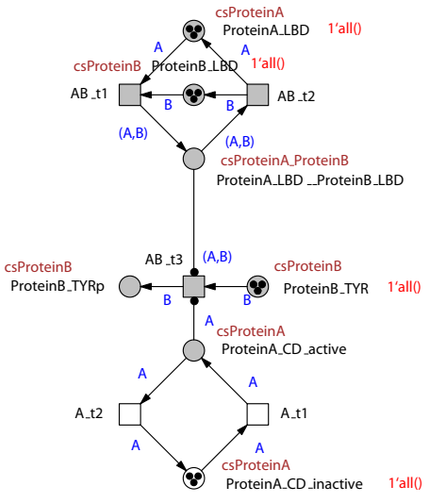
12 csProteinC ProteinC_X 4'all()

12 csProteinC ProteinC_Y 4'all()

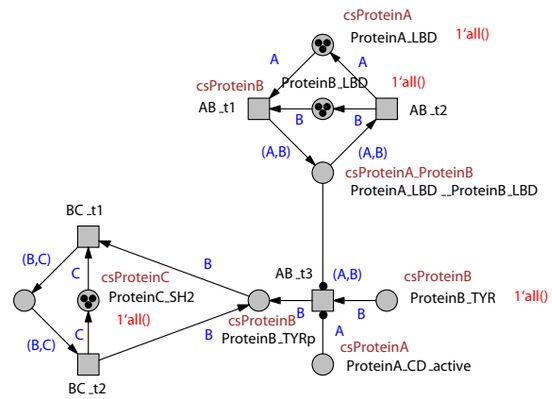
Mary-Ann Blätke, Christian Rohr - BioPPN 2015

Applying Coloured Petri Nets

Module of Protein A

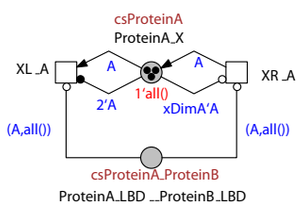


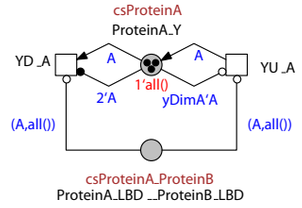
Module of Protein B



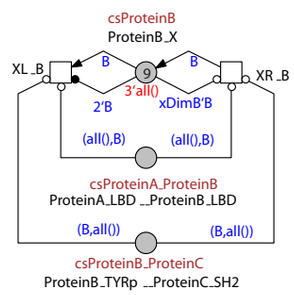
csProteinB_ProteinC
ProteinB_TYRp__ProteinC_SH2

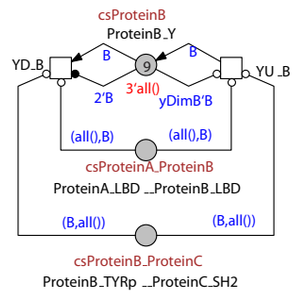
Movement of Protein A



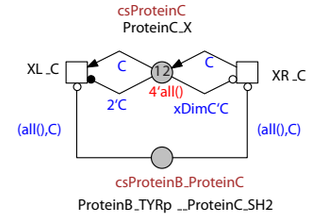


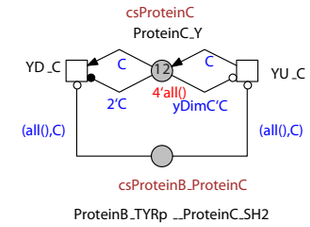
Movement of Protein B



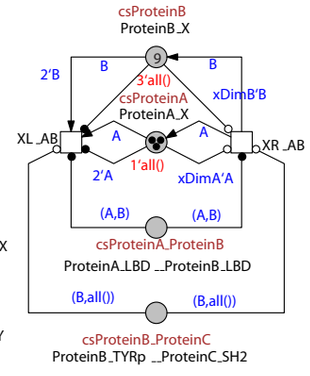


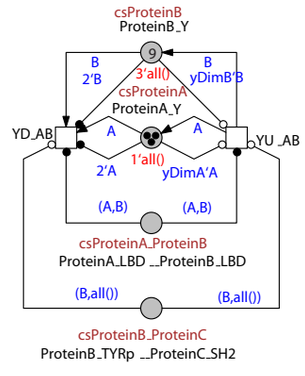
Movement of Protein C



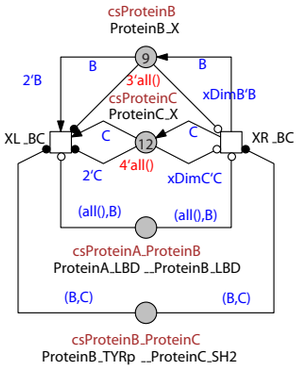


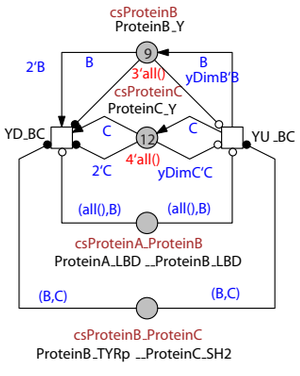
Movement of complex ProteinA_ProteinB



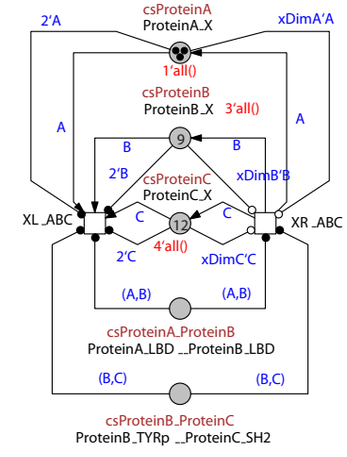


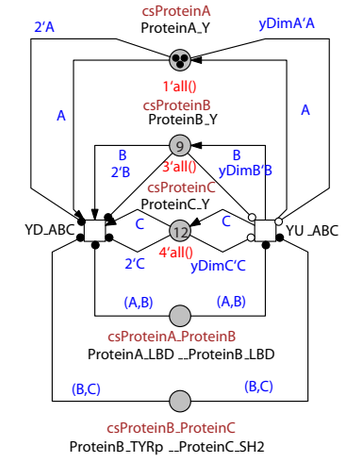
Movement of complex ProteinB_ProteinC



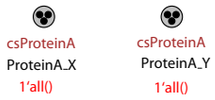


Movement of complex ProteinA_ProteinB__ProteinC

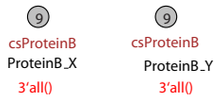




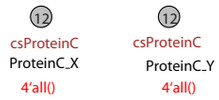
XY-Position of Protein A



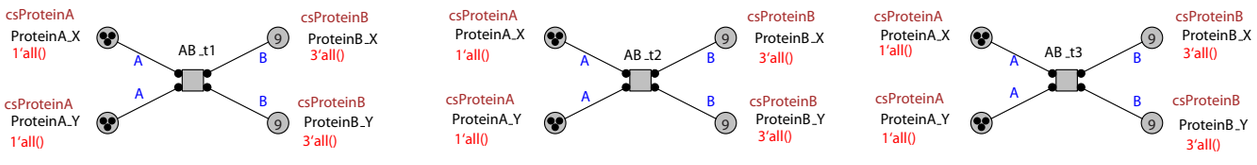
XY-Position of Protein B



XY-Position of Protein C



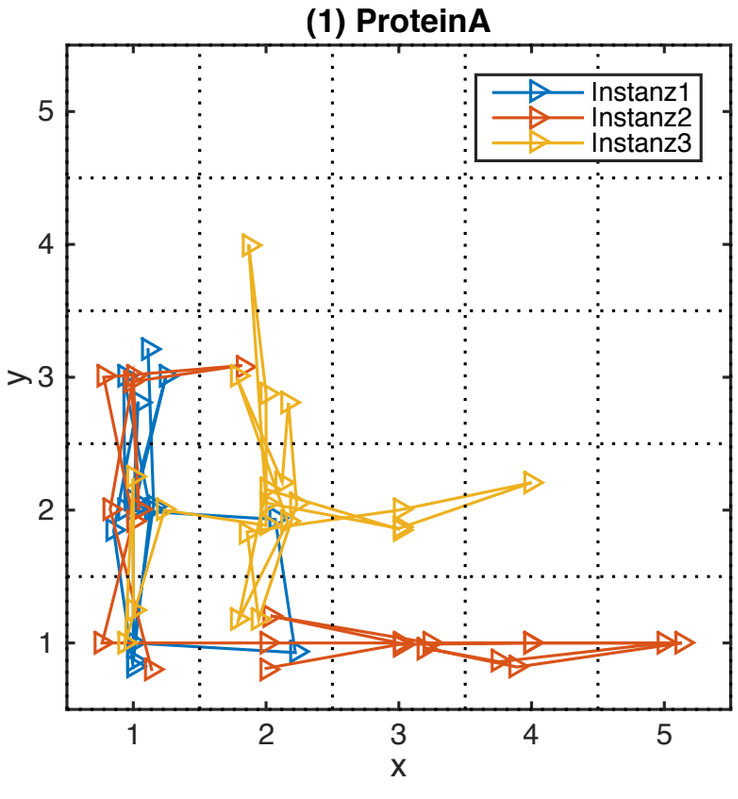
Restricted Interaction - Only if ProteinA (X,Y) = ProteinB (X,Y)

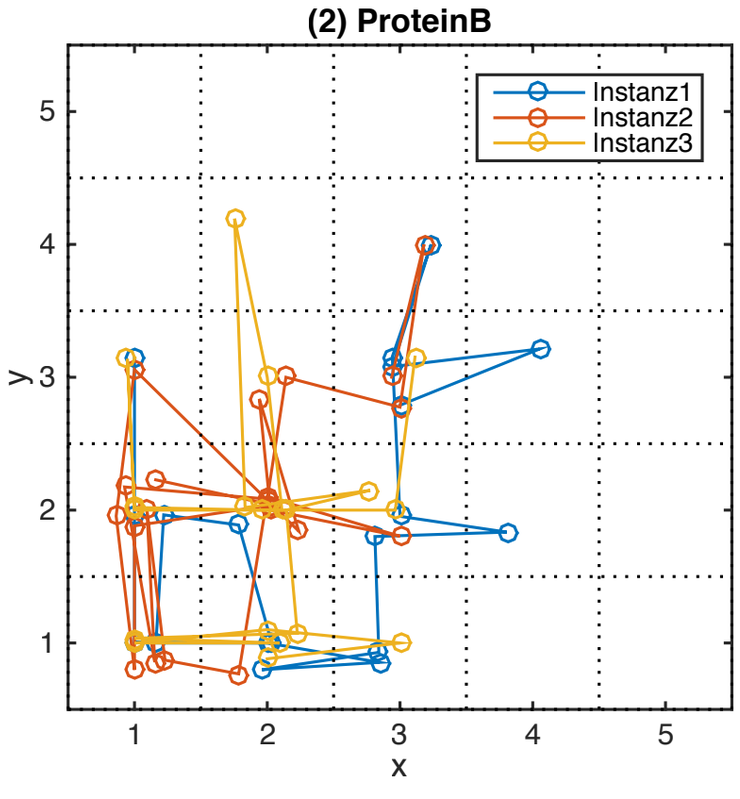


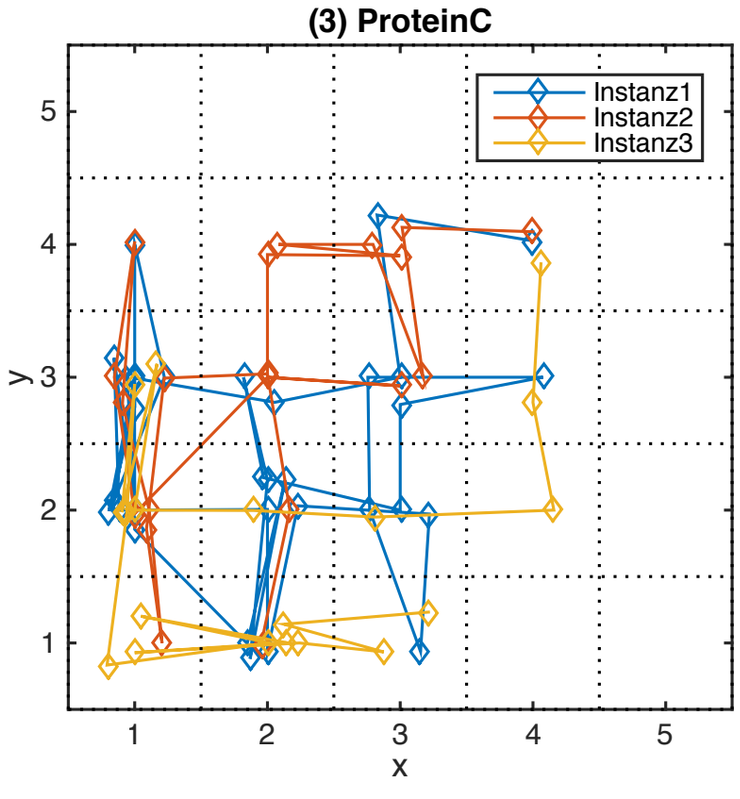
Restricted Interaction - Only if ProteinB (X,Y) = ProteinC (X,Y)

Stochastic Simulation

- Model parameter:
  - 5x5 2D-Grid for each component
  - 3 instances per component
- Model size:
  - |P|=57, |T|=267, |A|=1992

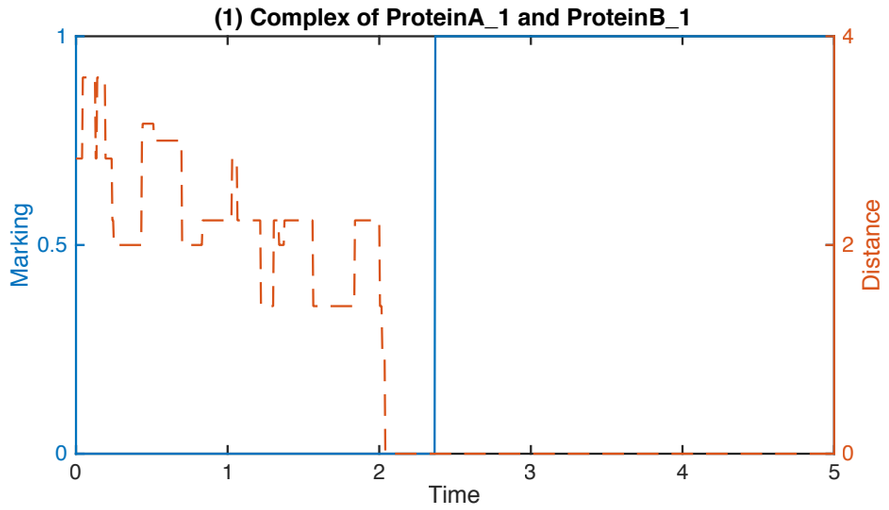


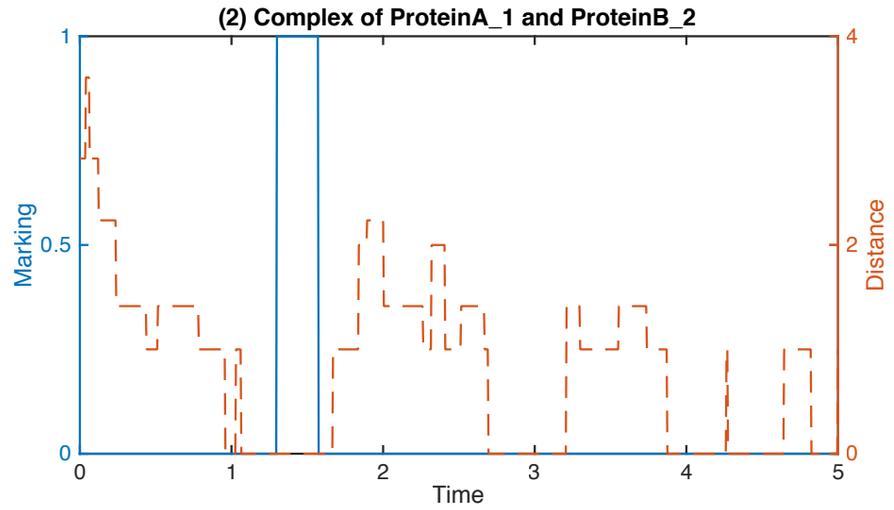


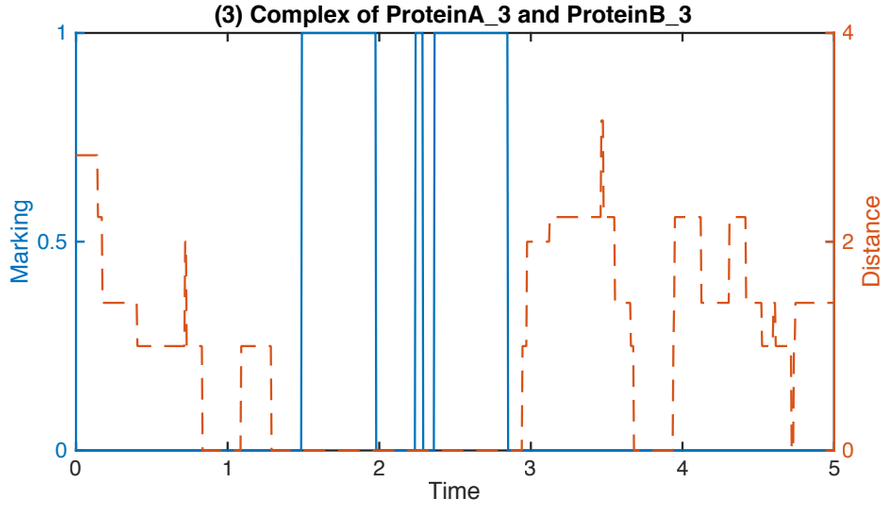


Mary-Ann Blätke, Christian Rohr - BioPPN 2015

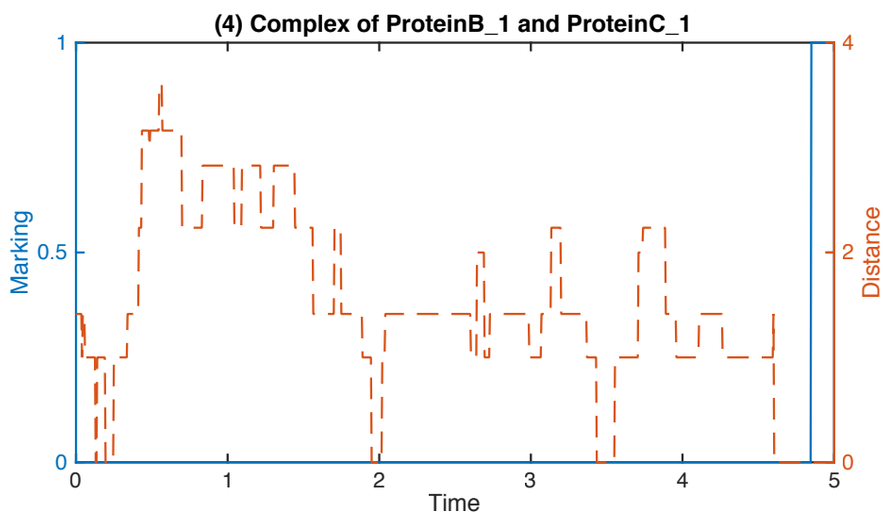
Stochastic Simulation

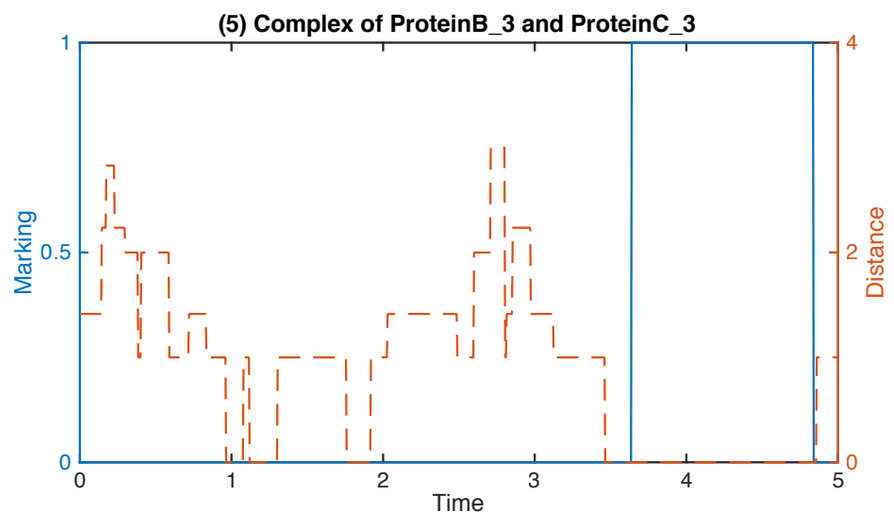






Components move as one entity while interacting





Summary

- Spatial Transformation Algorithm

    1. Explicit encoding of local positions
    2. Local restriction of interactions
    3. Explicit encoding of local position changes
    4. Explicit encoding of component instances using coloured Petri nets

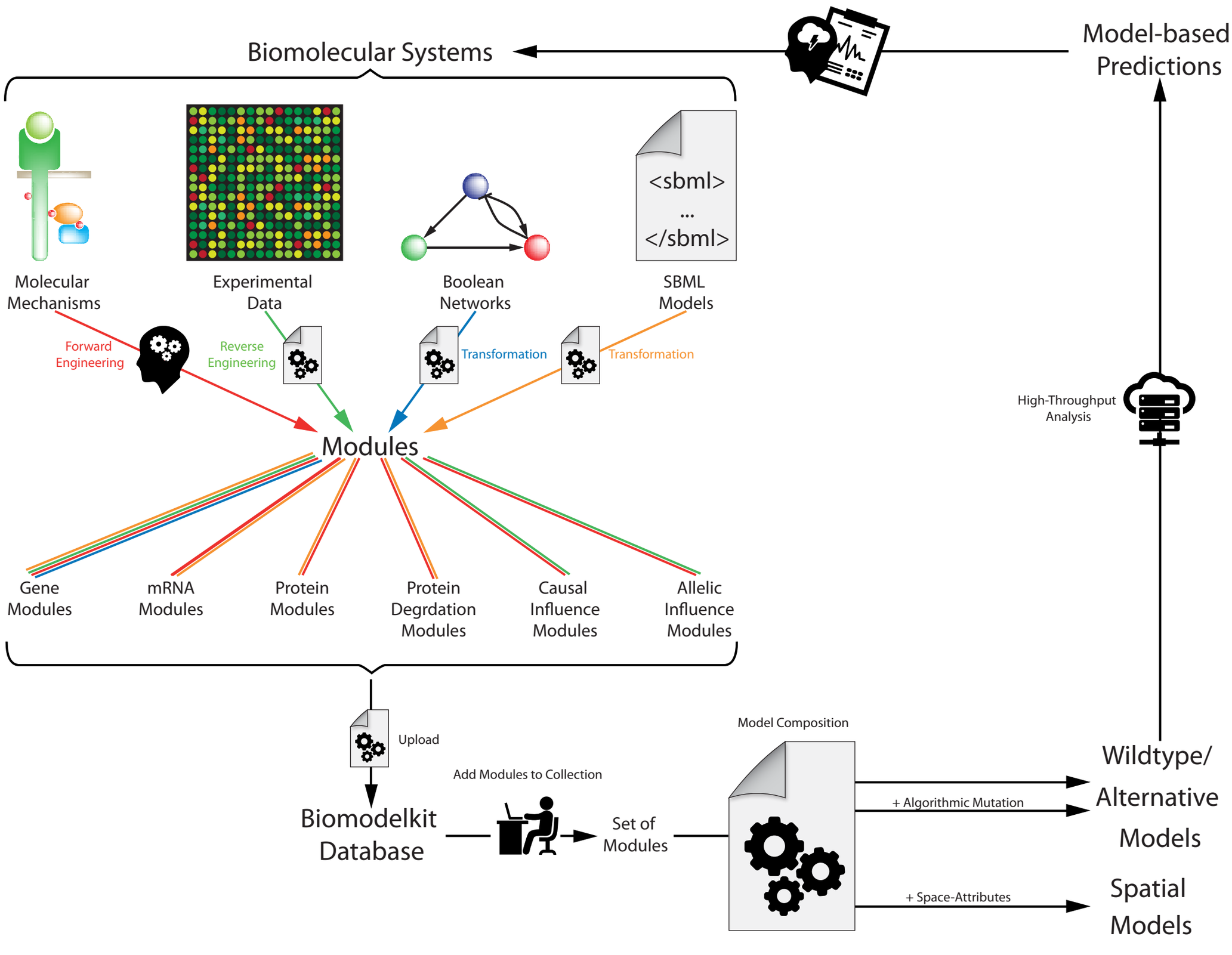
- No interference with the model structure

    ➡ Transformation is reversible

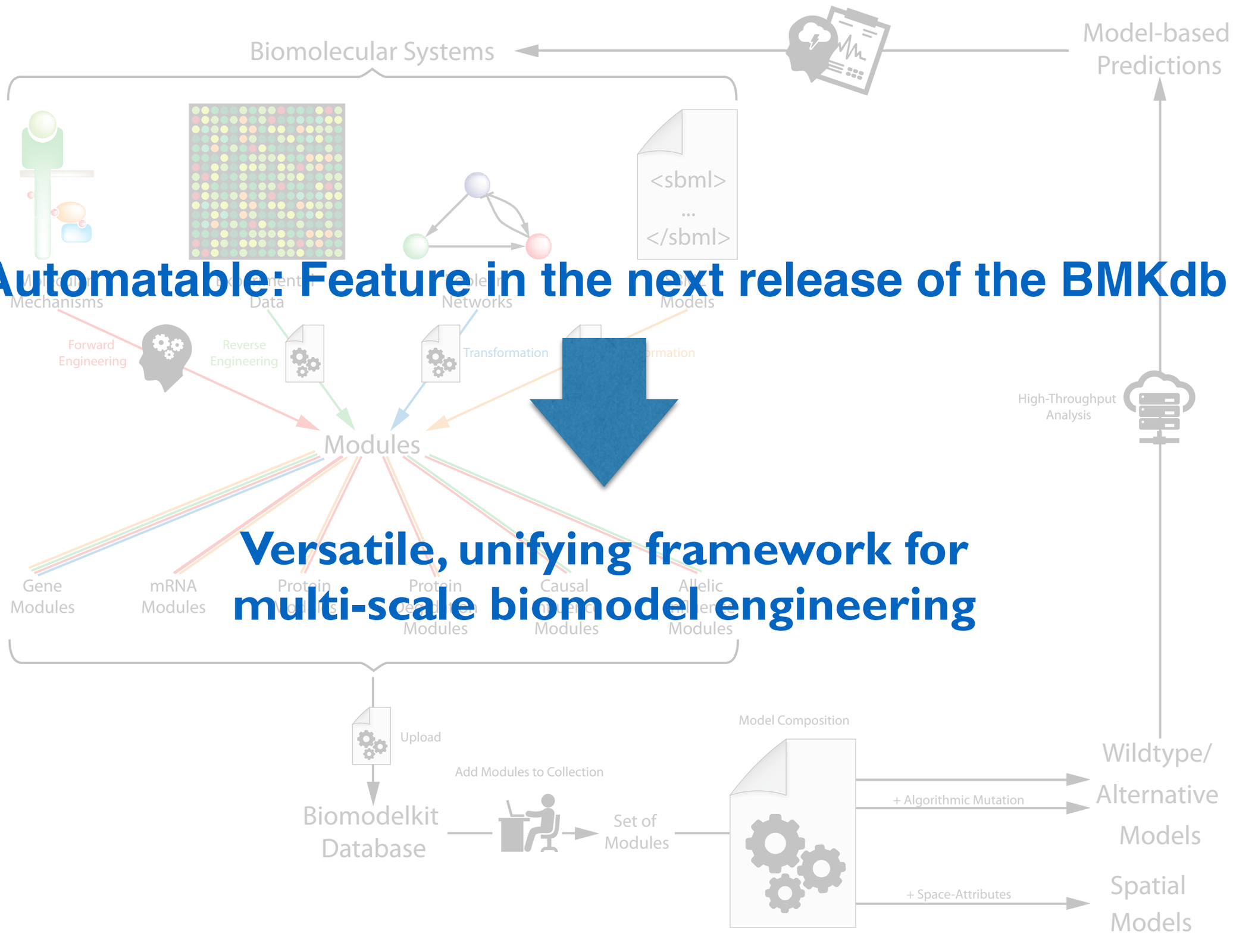
- Not restricted to discrete space

    ➡ Representation of continuous space by the use of continuous places

Summary

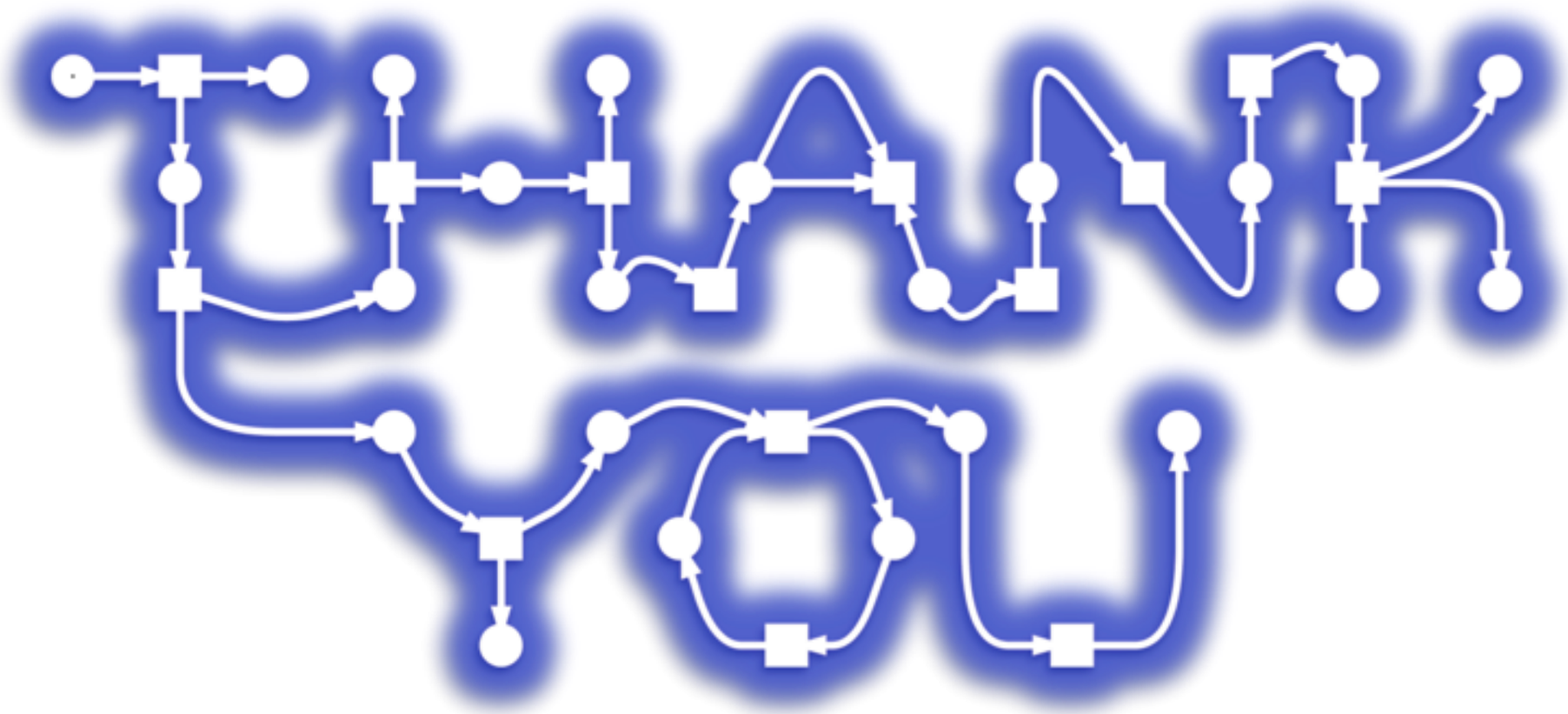

Biomolecular Systems
Model-based Predictions
Molecular Mechanisms
Experimental Data
Boolean Networks
SBML Models
<sbml>
...
</sbml>
Forward Engineering
Reverse Engineering
Transformation
Transformation
Modules
High-Throughput Analysis
Gene Modules
mRNA Modules
Protein Modules
Protein Degrdation Modules
Causal Influence Modules
Allelic Influence Modules
Upload
Biomodelkit Database
Add Modules to Collection
Set of Modules
Model Composition
Wildtype/ Alternative Models
+ Algorithmic Mutation
+ Space-Attributes
Spatial Models

Summary



Automatable: Feature in the next release of the BMKdb

Versatile, unifying framework for multi-scale biomodel engineering



Cooperation Partners

- Monika Heiner and Co-Workers, BTU Cottbus
- David Gilbert, Brunel University London
- Fred Scharper and Co-Workers, OvGU Magdeburg
- Tim Hucho, University of Cologne
- Fei Liu, Harbin Institute of Technology

Projects

- Consortium „Modelling of Pain Switches" 2009-2011
- Consortium „NoPain" 2013-2015

Graduate School

- IMPRS Magdeburg

Sachsen-Anhalt

Federal Ministry of Education and Research

FORSYS
Research Units for Systems Biology

Magdeburg Centre for Systems Biology
MaCS

b-tu
Brandenburg University of Technology Cottbus

IMPRS ProEng
International Max Planck Research School for Advanced Methods in Process and Systems Engineering
Magdeburg

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

DFG Deutsche Forschungsgemeinschaft

Recent Publications

Springer

Chapter 6

A Petri-Net-Based Framework for Biomodel Engineering

Mary Ann Blätke, Christian Rohr, Monika Heiner, and Wolfgang Marwan

Abstract Petri nets provide a unifying and versatile framework for the synthesis and engineering of computational models of biochemical reaction networks and of gene regulatory networks. Starting with the basic definitions, we provide an introduction into the different classes of Petri nets that reinterpret a Petri net graph as a qualitative, stochastic, continuous, or hybrid model. Static and dynamic analysis in addition to simulative model checking provide a rich choice of methods for the analysis of the structure and dynamic behavior of Petri net models. Coloring of Petri nets of all classes is powerful and for the representation of location and space in reaction networks since it combines the concept of Petri nets with the computational mightiness of a programming language. In the context of the Petri net framework, we provide two most recently developed approaches to biomodel engineering, the database-assisted automatic composition and modifica- tion of Petri nets with the help of reusable, automatic reconstruction of networks based features the framework provides multiple o context of systems and synthetic biology.

Keywords Automatic network reconstructio systems modelling · Modular modelling · Pe networks · Reverse engineering

M.A. Blätke · W. Marwan (✉)
Otto-von-Guericke-Universität Magdeburg, Universi
e-mail: wolfgang.marwan@ovgu.de

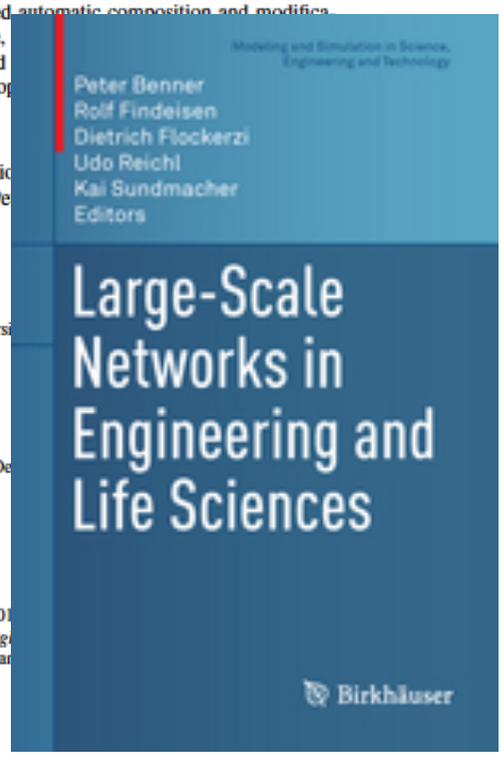
M.A. Blätke
e-mail: mary-ann.blaetke@ovgu.de

C. Rohr · M. Heiner
Brandenburg University of Technology, Platz der De

C. Rohr
e-mail: christian.rohr@b-tu.de

M. Heiner
e-mail: monika.heiner@b-tu.de

P. Benner et al. (eds.), *Large-Scale Networks in Engi*
Modeling and Simulation in Science, Engineering a
DOI 10.1007/978-3-319-08437-4_6



Chapter 7

ELSEVIER

BioModel Engineering with Petri Nets

Mary Ann Blätke[1], Monika Heiner[2] and Wolfgang Marwan[1]
[1]*Otto-von-Guericke University, Magdeburg, Germany,* [2]*Brandenburg University of Technology, Cottbus, Germany*

7.1 INTRODUCTION

BioModel Engineering

Systematically constructing, maintaining, and deploying artifacts are typical attributes of sound engineering, independent of the application field. Along these lines, BioModel Engineering is the science of engineering computational models of biochemical processes in an efficient, sustainable, and trustworthy manner.

In *Systems Biology*, models are used to describe our abstract understanding of biochemical processes and to predict their behavior, for example, in response to perturbations like mutations, chemical interventions, or changes in the environment. In *Synthetic Biology*, models support the reliable design and redesign of molecular networks and may serve as design templates for models have high explanatory and predictive power

In this chapter, we will explore how this am the unifying power of Petri nets, serving as umb continuous, and hybrid paradigms. An introductio transduction pathways can be found in [1]; a more ri given in [2].

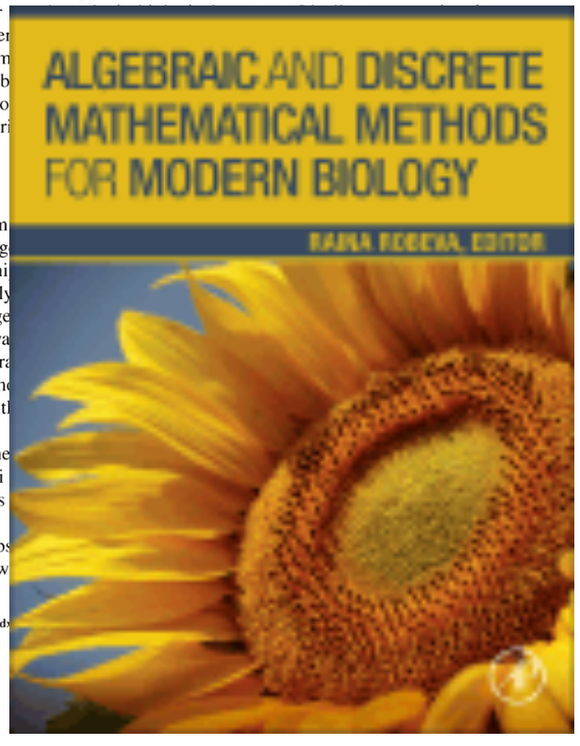
Petri Nets

The basic ideas of Petri nets, as we understand them in 1962 [3]. Petri nets are basically a formal languag semantics. They permit the modeling of relationshi transitions in the Petri net terminology and typically interpreted as any (bio-)chemical species, such as ge whereas events model (bio-)chemical reactions at va regulatory activation and deactivation, transport, tra stoichiometry of (bio-)chemical reactions or the am structure of a Petri net is unambiguous and explicitl concurrency.

Enhancing these key modeling principles by the from programming languages, yields colored Petri "discrete data type." They are particularly strong, as similar processes in similar components.

Petri nets may represent species on different abs multicellular organisms and populations. This allow

Algebraic and Discrete Mathematical Methods for Modern Biology. http://dx



Scaling Properties

- The unfolded model scales with:
  - number of axes (1D, 2D, 3D)
  - number of instances and the resulting number of complexes among the components

Number of Instances	n = 1	n =5	n = 10	n = 50	n = 100
Places	15	115	330	5650	21300
Transitions	31	895	5440	533200	4131400
Arcs	160	6880	42760	4253800	33007600
Unfolding Time (s)	**0.124**	**0.305**	**1.536**	**180.763**	**1608.802**

CPU: 2 x 2.66 GHz 6-Core Intel Xeon, RAM: 24 GB 1333 MHz DDR3 ECC, OS: MacOS X 10.10, Software: Snoopy (IDD-based unfolding)