The Microsoft Research - University of Trento
Centre for Computational and Systems Biology

Technical Report CoSBi 04/2008

Automated generation of narrative language programs from NCI-Nature's Pathway Interaction Database

Filippo Polo
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
`filpolo@tin.it`

Corrado Priami
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
`priami@cosbi.eu`

UNIVERSITÀ DEGLI STUDI DI TRENTO

The Microsoft Research - University of Trento
Centre for Computational and Systems Biology

Second Level International Master in Computational and Systems Biology

---

Master Thesis

Automated generation of narrative language programs from NCI-Nature's Pathway Interaction Database

Advisor
Corrado Priami

Student
Filippo Polo

Introduction

In recent years, the field of computational biology has undergone a rapid growth, with great amounts of research focused on applying computer science to the field of biology. One of the main areas of this discipline deals with building computer models of biological entities and phenomena, in order to perform formal analyses of these models, or to run computer simulations, or both.

These so-called *in silico* experiments allow the researcher to model in seconds reactions that would take a long time in the real world, to easily change the parameters of the system to see what changes result, and to know the details of the state of the system at all times. With these tools, the computer model can be used to generate new hypotheses to be tested with *in vitro* and/or *in vivo* experiments.

One of these languages which is gaining moment is *beta binders*. This language is based on stochastic pi-calculus, a formal language which was first developed to study the behavior of parallel systems where a number of processes exchange messages. By considering molecular interactions in these terms, we can use beta binders to create models and run computer simulations of said interactions. Papers describing the beta binders language and tools to develop models with it are available from the Centre for Computational and Systems Biology web site [2] [3].

A computer simulation, however, can only resemble reality within the limits of the model which is being executed. The creation of a good computer model remains a relatively hard task, especially for biologists who typically aren't used to employing formal languages in their research. On the other hand, computer scientists seldom have the biological knowledge required to ensure that a model is actually close to reality.

For this reason, there is currently a dearth of detailed, accurate computer models of actual biological pathways, which could be used to both make *in silico* experiments related to that pathway, and to advance the development of simulators and modeling languages themselves.

Maria Luisa Guerriero has described a *narrative language* [1] which is designed to allow the user to enter descriptions of biochemical events in a form which closely resembles the natural language used by biologists. However, the narrative language can be automatically translated to beta binders, which allows for these events to be converted into a working computer model.

The Pathway Interaction Database (PID) [4], maintained by the National Cancer Institute and Nature, is a large online database of biochemical interactions, which the PID web site uses to display graphs representing interactions within biological pathways. This database is stored in a format which shares many features with a narrative language program – in fact, it contains almost all the information needed to make a narrative language program, except for the quantitative data (such as reaction rates, molecule quantities and so on).

We aim to create a tool that allows the user to select interactions from the PID, add quantitative data, and automatically generate a narrative language program modeling the interactions. Using Guerriero's tool, the user will then be able to create a beta binders model. Ultimately, this should allow a user to quickly and easily make executable models out of PID interactions.

In the following sections, we will briefly describe the narrative language and the PID database, outline their similarities and key differences, and describe the PIDconvert tool we've developed.

The narrative language

The narrative language was developed in 2007 by Maria Luisa Guerriero, with the intent of designing a language which would be as friendly as possible to non-technical users while still being formal enough to be converted into beta binders and used to run a computer simulation.

The user provides data on the modeled entities, the reactions and the cell compartments where these reactions take place, and then describes biochemical events in a semi-formal, textual representation. A typical narrative language program can be represented in the form of tables. We will provide an example of a narrative language program in this form.

Compartments

Table 1 contains the data for several cell compartments. The number of dimensions can be either 2 or 3; cell volumes have three dimensions, while membranes usually have two dimensions.

Table 1 - narrative language compartments

Id	Name	Size	Unit of measure	Number of dimensions
1	Exosol	$9.91 \cdot 10^{-12}$	l	3
2	CellMembrane	$1.26 \cdot 10^{-7}$	dm^2	2
3	Cytosol	$2.09 \cdot 10^{-12}$	l	3
4	Nucleus	$0.25 \cdot 10^{-12}$	l	3

Components

Table 2 contains the data for some molecular components. The list of “sites” allows the user to detail where an interaction can modify the molecule. An arbitrary number of sites can be defined, each with a state that can be active or inactive. States can also be associated to the component itself, rather than a site. The “compartment” column describes where the component can be, and whether it will be in that compartment in the initial state.

The initial quantity can be specified, along with a percentage which expresses how reliable the quantity information is. This type of expression is used for most other numerical values in the language.

Table 2 - narrative language components

Name	Descr.	Site	Site state	Site active	State	State active	Comp.	Comp. active	Initial qty	Rel.ty
LIF	Ligand				Bound	False	1	True	3000	100%
gp130	Receptor	LIF	Bound	False	Bound	False	2	True	1000	50%
		Y767	Phospho	False	Dimer	False				
		Y814	Phospho	False						
STAT3	Effector	Y705	Phospho	False	Dimer	False	3	True	5000	30%
		gp130	Bound	False			4	False		

Reactions

Table 3 contains the data for some reactions. This includes the reaction rate constant, and the reaction volume. Both can be specified along with their unit of measure and the reliability value.

Table 3 - narrative language reactions

Id	Type	Rate	Unit	Rel.ty	Reaction volume	Unit	Rel.ty
1	Binding	$4.8 \bullet 10^{7}$	$M^{-1}Min^{-1}$ *(k_a)*	80%	$9.91 \bullet 10^{-12}$	l	50%
2	Unbinding	$3.6 \bullet 10^{-1}$	Min^{-1} *(k_{off})*	80%	$9.91 \bullet 10^{-12}$	l	50%
3	Dimerization	Inf	$M^{-1}Min^{-1}$	50%	$9.91 \bullet 10^{-12}$	l	50%
4	Phosporylation	12	Min^{-1} *(k_{cat})*	80%	$9.91 \bullet 10^{-12}$	l	50%
5	Homodimerization	Inf	Min^{-1}	50%	$2.09 \bullet 10^{-12}$	l	50%
6	Relocation	10	Min *($t_{1/2}$)*	10%	$2.09 \bullet 10^{-12}$	l	50%

Narrative

Table 4 is the key part of the program – the detailing of the events comprising the model. This table references data from the previous tables to construct a narrative of the reactions being modeled.

The events are grouped into processes. For each event, the corresponding reaction is specified, and optionally an event which is alternative (mutually exclusive) to this one. The event itself is a semi-formal sentence, describing an action which can happen if some conditions are met.

Table 4 - narrative language events

Id	Description	R	A
LIF-gp130 binding			
1	if gp130.LIF is not bound and LIF is not bound then LIF binds gp130 on LIF	1	
2	if gp130.LIF is bound and LIF is bound then LIF unbinds gp130 on LIF	2	
LIF pathway			
3	if gp130.LIF is bound and LIFR.LIF is not bound then LIFR dimerizes with gp130	3	2
4	if gp130 is dimer then gp130 phosphorylates on Y767;Y814	4	
STAT3 pathway			
5	if gp130.Y767 is phospho and STAT3 is in 3 then gp130 binds STAT3 on gp130	1	
6	if STAT3.gp130 is bound then STAT3 phosphorylates on Y705	4	
7	if STAT3.gp130 is bound and STAT3.Y705 is phospho then gp130 unbinds STAT3 on gp130	2	
8	if STAT3.Y705 is phospho and STAT3.gp130 is not bound then STAT3 homodimerizes	5	
9	if STAT3 is in 3 and STAT3 is dimer then STAT3 relocates to 4	6	

These tables are represented in a textual form, respecting the narrative language syntax, which can be passed to the narrative language compiler for translation into beta binders. This textual form of a narrative language program is going to be the output of our tool.

The Pathway Interaction Database

The Pathway Interaction Database, or PID for short, is a database of biochemical pathways, maintained by the National Cancer Institute and Nature. At the moment of writing, the database contains 3726 interactions which have been curated by NCI-Nature, and 5833 which have been imported from the BioCarta [5] and Reactome [6] databases. Data in the PID is provided alongside evidence codes and PubMed article references related to each interaction.

The PID is accessible on the Internet at the address http://pid.nci.nih.gov/PID/. The web interface allows users to query the database for molecules or pathways. Sections of the database can be visualized as graphs, where nodes represent molecules and interactions. An edge going from an interaction to a molecule indicates that the molecule is a product of that interaction. Edges going from a molecule to an interaction can represent an input of that interaction, an enzyme or catalyst (called an agent), or an inhibitor, depending on the edge color. Graphs produced by the PID web interface can be clicked, linking to detailed information about the selected interaction or molecule instance.

When using the PID, an important distinction must be made between molecules and molecule *instances*. We refer to a molecule when we are discussing properties of the molecule in general, while we refer to a molecule instance when we are discussing a *specific* molecule, such as the one that is taking part to a given interaction, or the one that is part of a given complex. Only molecule instances can have state information.

Molecules and interactions are the main components of the database.

Molecules

A molecule can include this data:

- One or more **names**.
- A molecule **type**, such as “protein”, “compound”, or “complex”.
- If the data entry is a molecule *family*, then it will have a list of all molecules in the family.
- If the molecule is a *complex*, then it will have a list of molecule instances, describing the complex components.

- If the molecule is a part of another molecule, it will have a reference to the larger molecule.

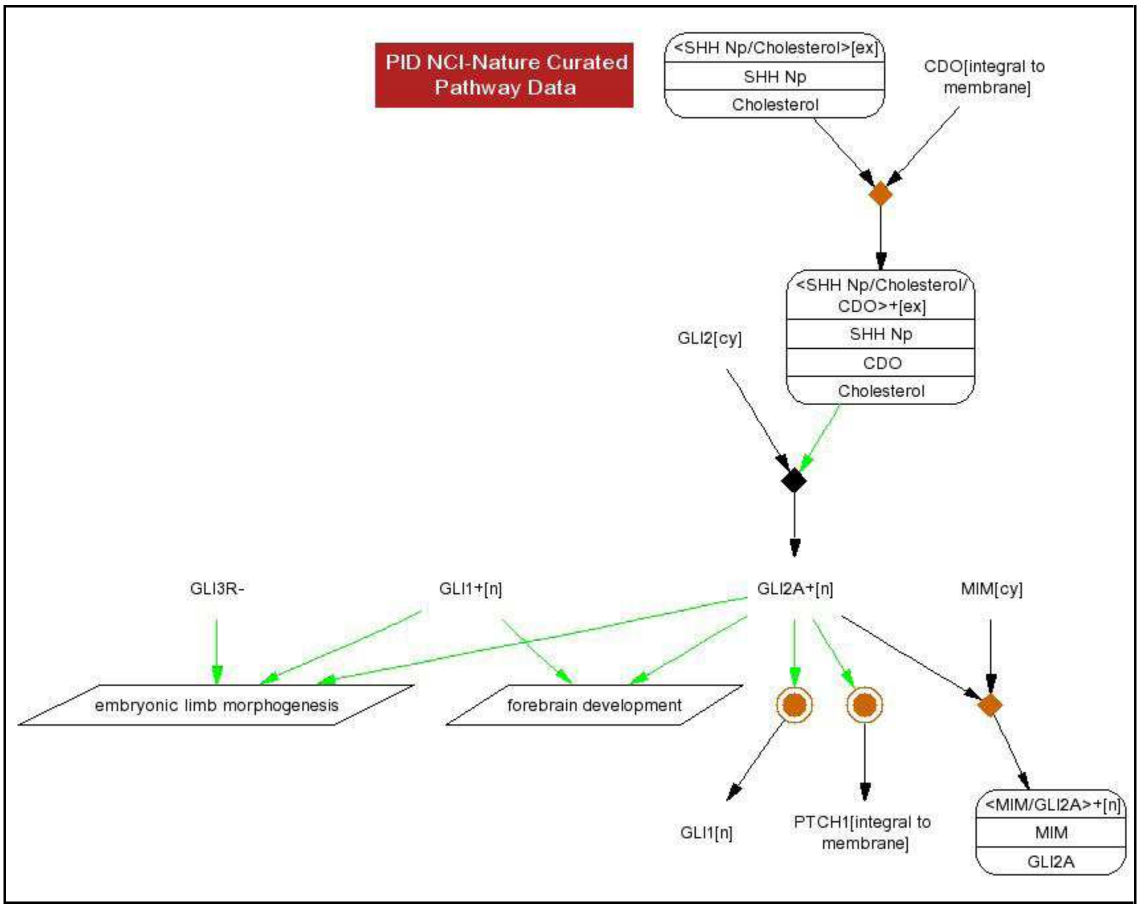
Notice that the molecule entries do not list binding sites; these are detailed in the interaction that uses them, as part of the molecule instance record.

Molecule instances

A molecule instance contains state information. It can include this data:

- A reference to the corresponding molecule entry.
- Optionally, a **location** such as nucleus or cytosol.
- Optionally, an **activity state** for the molecule. Typically, this is either *active* or *inactive*, but it may also be *active2* or *active3* as some molecules can be active in several different ways.
- Optionally, a list of **post translational modifications**, which describe the site and the type of modification (e.g. phosphorylation).

Figure 1 - typical PID representation of some biochemical interactions



Interactions

Interactions can contain this data:

- An interaction **type** such as *translation* or *modification*.
- Optionally, a reference to the **pathway** to which the interaction belongs.
- Optionally, evidence codes describing how the interaction data was obtained.
- Optionally, references to PubMed articles related to the interaction.
- A list of molecule instances, which are the interaction components. Each of these can be one of:
    - **Input**: this molecule is consumed or modified by the reaction.
    - **Output**: this molecule is produced by the reaction, or is the modified input.
    - **Agent**: this molecule takes part in the reaction but is not itself changed.
    - **Inhibitor**: this molecule inhibits the reaction.

The PID also contains information related to pathways, but we won't make use of that in this thesis.

Note that PID doesn't feature any quantitative data – there are no reaction rates or molecule quantities, as representing this type of information is not the database's objective. This means that in order to automatically generate a computer model out of PID data, the user will somehow have to enter these values himself.

Key similarities and differences

Our aim is to create a tool that automatically generates narrative language programs from the PID database. Clearly then, every similarity between PID and the narrative language will help us, while each difference will be an obstacle we need to overcome. It is therefore crucial to identify the ways in which the two formalisms are similar and different.

First of all, the obvious difference is that one is a programming language and the other is a database. Regardless, both express *models*, and this is the point of view we need to employ. The narrative language has a strong component of data representation (it can even be displayed as tables), and PID interactions represent events, so there is more similarity than one might assume.

Compartments / locations

The concept of where an entity is located in the cell is called a **compartment** in the narrative language, and a **location** in PID. While the narrative language explicitly defines compartments, the PID only references their name and doesn't define a specific data type for them.

We generate a compartment declaration for the narrative language for each location used by the interactions selected by the user. The user has to supply the compartment size and number of dimensions.

Components / molecules

The PID refers to entities as **molecules**, while the narrative language uses the more generic term **component**. Both define the concept of the state of an entity, and both can represent sites, but there are significant differences.

The narrative language can represent multiple states for a component, such that a component can be bound, active, phosphorylated and so on, or any combination of these states. However, each state is binary; the component either is in that state, or it isn't.

By contrast, the PID only has a single state variable for a molecule instance, but this state is not binary. It can be active or inactive, but it is frequently unspecified and sometimes it can be “active2” or “active3”.

Also, in the narrative language we declare states when we define the component; in the PID, the molecule itself doesn't have any state declaration because state information is stored in the molecule instance, within the interactions or complexes that use that molecule.

PID sites are strictly sites for post translational modifications. Actual molecule binding is either expressed as a complex, or not expressed at all: for example, enzyme binding to substrate is typically not represented, and the process is expressed as a single interaction where the substrate is an input, the product is an output, and the enzyme is an agent.

The narrative language has general-use sites, but it doesn't have an explicit construct for complexes. Typically, one defines a complex by defining binding sites in one of the molecules for the other molecules in the complex. This type of representation is asymmetrical – that is, if two molecules form a complex, then one has a binding site for the other. By contrast, the PID represents complexes as separate molecules which contain molecule instances; the relation between these molecule instances is simply that they are part of the same complex.

Similarly to the way state declarations are handled, the narrative language requires declaration of sites when we declare the component, while the PID only defines sites when they are actually used, within molecule instances.

In order to generate component declarations for the narrative language, we start with the PID molecule entry and the molecule instance entries in each interaction selected by the user. The name has to be rewritten in order to exclude special characters, but the most complex operations deal with states and sites.

We look for states and sites in the molecule instances used by the interactions. For states, if any interaction specifies a state for the molecule instance, then we add an "active" state to the component corresponding to the molecule. This approach can handle unspecified states simply by not using the state value in conditions for the interaction, but it can't handle usage of "active2" or "active3".

For sites, we look for post translational modifications in the molecule instances, and add a site for each of them. The site's state will be the PTM type, and the site name will be the amino acid's letter followed by its position on the protein.

As explained later in the “interactions” section, we will also generate several additional sites to deal with agent binding.

Finally, we add a compartment entry for each “location” field found in a molecule instance.

At this point, the user will have to supply quantitative information, such as the starting amount of molecules, initial states and so on.

At present, we don’t deal with the formation and unbinding of complexes, so we simply treat them as if they were a single molecule.

Reactions / interactions

The greatest differences are between the ways interactions are managed in the PID and in the narrative language. As we will explain, there are two fundamental differences.

The first and most problematic main difference is that, by virtue of the way it’s structured, the PID expresses interactions by describing the state of the system before the interaction, and the state of the system after the interaction. It does not describe what happens, or how the state changes. By contrast, the narrative language may declare the state of the system before the interaction (as conditions), but then it describes the events that happen to the system.

For example, a typical PID interaction entry may have a molecule as an input, and the same molecule – now active – as an output. On the other hand, the corresponding narrative language event will be in the form of “the molecule activates”, with a condition that the molecule must be inactive.

It is trivially easy to infer from the PID form that the event that happens is the activation of the molecule. The problem is that the inference quickly becomes harder as the interaction becomes more complex. Consider the case in which a phosphate group moves from a molecule to another, with the aid of one or more agents, subject to an inhibitor. The PID representation might simply list one input molecule and a different output molecule as being active, and list agents and inhibitors.

There is nothing telling us which input binds to which agent or inhibitor. There is nothing telling us whether first an input activates and then another deactivates, or the opposite. We don’t know if the inhibitor and the agent bind to the same site, or to different sites. Also crucially, there is no way to represent this interaction as a single event in the narrative language – we must create multiple events, and somehow make them atomic.

Unfortunately, some of this information is simply not there. The PID does not model agent binding to inputs, and neither it models the sequence of events which comprise an interaction. We have to make assumptions which result in the correct interaction, for example by assuming that all agents bind to the first specified input. This will probably not correspond to the biological reality in most cases.

Tying multiple events together in a single interaction is also a challenge. The structure we use is as follows.

First, we define a list of conditions that have to be satisfied before the state changes can happen. Consider an interaction where an agent activates a molecule. The condition is that the input must be inactive, and the agent must be bound. During this step, we may have to add some binding sites to an input molecule, where we're going to bind agents.

Then, we define events that setup the required conditions. In the example, we define an event in which the agent binds to the input, subject to the condition that the input must be inactive.

Then, we define the events that result in the state transition. In the example, we define an event where the input activates, subject to the conditions we determined.

Then, we define conditions that verify that the state transition has occurred. In the example, we define the condition that the input is active.

Finally, we define "cleanup" events that have to happen after the state transition has occurred. In the example, we define an event where the agent unbinds from the molecule, subject to the condition that the state transition has occurred.

We have expanded this approach to deal with translocations and post translational modifications as well (but not inhibitors). However, there is still a large section of interactions that can't be handled in this way, mainly the interactions where a molecule seems to transform into another molecule entirely. Most of these interactions arise from the fact that we don't handle complexes, so every interaction where a complex is formed or unbound looks like an interaction where a molecule changes into another. However, there is also a number of interactions where the PID actually shows an input line with a molecule, and an output line with another molecule, neither of which are complexes. The narrative language doesn’t have a construct to represent a transformation event, so handling these cases is difficult.

The second big difference is that the narrative language distinguishes between a reaction and the corresponding events. This simply means that quantitative information such as the reaction rate and volume is described in a reaction declaration, which is separate from the narrative of events. In the PID, there is no quantitative information, so the interaction entries contain the entire available information on each interaction.

As we have seen, most PID interactions are going to result in multiple events in the narrative language. Each event must have an associated reaction entry. Therefore, we create a reaction declaration for each event we create. The user will have to specify the reaction rates for each of them, depending on the information he has available.

Generally, only one of these reactions (such as the one where the agent binds to the input) will have a reaction rate, and the others will have an infinite rate. This is because the events which have been generated from one interaction are supposed to be atomic, so once the conditions are met, the state transition should immediately take place. However, the user might want to model the various sub-steps of an interaction; in this case, he could assign a valid rate to two or more reactions.

Notice that we are not modeling the possibility for an agent to unbind before the state transition happens. In the real chemical world, enzymes can and do bind and then unbind without starting the reaction. The choice not to model that was made to adhere more closely to the PID model, in which all interactions are atomic.

For similar reasons, we don’t automatically add reverse reactions. Overall, we are trying to generate models which reflect the PID interaction as closely as possible. If a reverse interaction is present in the PID, and the user wishes to model that, he should explicitly select it for the generation of the narrative language program.

Processes

The narrative language allows the definition of processes, which are groups of events. This grouping is not crucial; our tool defines a process for each interaction, containing all the events which have been generated to represent it.

The PIDconvert tool

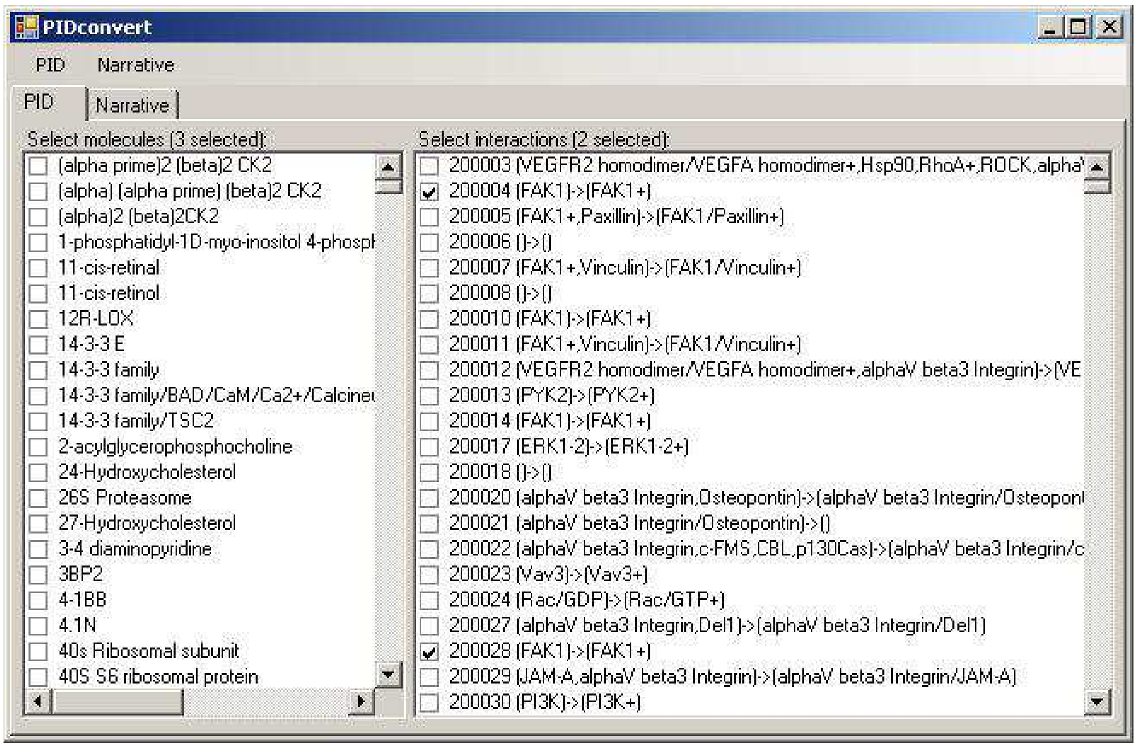
PIDconvert is a Windows program, developed in C#. It requires the **NCI-Nature_Curated.xml** file, which contains the main PID database and can be downloaded from the PID web site.

The tool's interface is divided in two main sections, called **PID** and **narrative**.

PID view

When PIDconvert is started, it will load the PID database and create a list of molecules and a list of interactions, both of which can be seen in the PID view of the tool.

Figure 2 - PIDconvert PID view



Select molecules (3 selected):

- ☐ (alpha prime)2 (beta)2 CK2
- ☐ (alpha) (alpha prime) (beta)2 CK2
- ☐ (alpha)2 (beta)2CK2
- ☐ 1-phosphatidyl-1D-myo-inositol 4-phospl
- ☐ 11-cis-retinal
- ☐ 11-cis-retinol
- ☐ 12R-LOX
- ☐ 14-3-3 E
- ☐ 14-3-3 family
- ☐ 14-3-3 family/BAD/CaM/Ca2+/Calcinet
- ☐ 14-3-3 family/TSC2
- ☐ 2-acylglycerophosphocholine
- ☐ 24-Hydroxycholesterol
- ☐ 26S Proteasome
- ☐ 27-Hydroxycholesterol
- ☐ 3-4 diaminopyridine
- ☐ 3BP2
- ☐ 4-1BB
- ☐ 4.1N
- ☐ 40s Ribosomal subunit
- ☐ 40S S6 ribosomal protein

Select interactions (2 selected):

- ☐ 200003 (VEGFR2 homodimer/VEGFA homodimer+,Hsp90,RhoA+,ROCK,alpha
- ☑ 200004 (FAK1)->(FAK1+)
- ☐ 200005 (FAK1+,Paxillin)->(FAK1/Paxillin+)
- ☐ 200006 ()->()
- ☐ 200007 (FAK1+,Vinculin)->(FAK1/Vinculin+)
- ☐ 200008 ()->()
- ☐ 200010 (FAK1)->(FAK1+)
- ☐ 200011 (FAK1+,Vinculin)->(FAK1/Vinculin+)
- ☐ 200012 (VEGFR2 homodimer/VEGFA homodimer+,alphaV beta3 Integrin)->(VE
- ☐ 200013 (PYK2)->(PYK2+)
- ☐ 200014 (FAK1)->(FAK1+)
- ☐ 200017 (ERK1-2)->(ERK1-2+)
- ☐ 200018 ()->()
- ☐ 200020 (alphaV beta3 Integrin,Osteopontin)->(alphaV beta3 Integrin/Osteopont
- ☐ 200021 (alphaV beta3 Integrin/Osteopontin)->()
- ☐ 200022 (alphaV beta3 Integrin,c-FMS,CBL,p130Cas)->(alphaV beta3 Integrin/c
- ☐ 200023 (Vav3)->(Vav3+)
- ☐ 200024 (Rac/GDP)->(Rac/GTP+)
- ☐ 200027 (alphaV beta3 Integrin,Del1)->(alphaV beta3 Integrin/Del1)
- ☑ 200028 (FAK1)->(FAK1+)
- ☐ 200029 (JAM-A,alphaV beta3 Integrin)->(alphaV beta3 Integrin/JAM-A)
- ☐ 200030 (PI3K)->(PI3K+)

The molecule list uses the molecule names from the database, but there is no name for interactions within the PID (only an ID). For the user's convenience, we generate a name from the names of the interaction's inputs and outputs, but this still results in some overlap; it is advisable to check the interaction ID using the PID web interface before using PIDconvert to generate the narrative language program.

The user can select any number of interactions in the rightmost list. These are the interactions which will be included in the narrative language program.

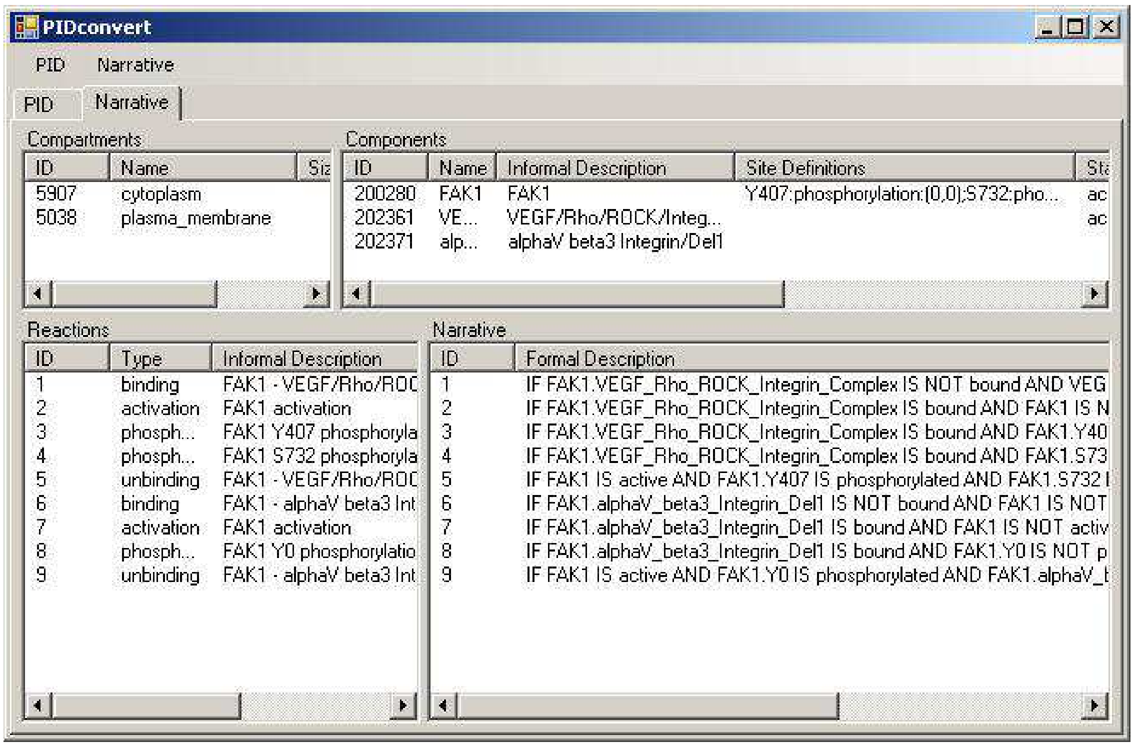
The user can also select molecules from the leftmost list, but this selection won't be directly used by the generator. However, the user can have the tool automatically select all interactions which involve the selected molecules (and vice versa). This includes molecules that are used as inputs, outputs, agents and inhibitors.

When the selection is done, the user can have the tool generate the declarations and narrative of events, and move to the narrative view.

Narrative view

The narrative view displays the generated program in table form, and allows the user to enter the required quantitative data before generating the program text.

Figure 3 - PIDconvert narrative view



Compartments

ID	Name	Siz
5907	cytoplasm	
5038	plasma_membrane	

Components

ID	Name	Informal Description	Site Definitions	Sta
200280	FAK1	FAK1	Y407:phosphorylation:(0,0);S732:pho...	ac
202361	VE...	VEGF/Rho/ROCK/Integ...		ac
202371	alp...	alphaV beta3 Integrin/Del1		

Reactions

ID	Type	Informal Description
1	binding	FAK1 - VEGF/Rho/ROC
2	activation	FAK1 activation
3	phosph...	FAK1 Y407 phosphoryla
4	phosph...	FAK1 S732 phosphoryla
5	unbinding	FAK1 - VEGF/Rho/ROC
6	binding	FAK1 - alphaV beta3 Int
7	activation	FAK1 activation
8	phosph...	FAK1 Y0 phosphorylatio
9	unbinding	FAK1 - alphaV beta3 Int

Narrative

ID	Formal Description
1	IF FAK1.VEGF_Rho_ROCK_Integrin_Complex IS NOT bound AND VEG
2	IF FAK1.VEGF_Rho_ROCK_Integrin_Complex IS bound AND FAK1 IS N
3	IF FAK1.VEGF_Rho_ROCK_Integrin_Complex IS bound AND FAK1.Y40
4	IF FAK1.VEGF_Rho_ROCK_Integrin_Complex IS bound AND FAK1.S73
5	IF FAK1 IS active AND FAK1.Y407 IS phosphorylated AND FAK1.S732 I
6	IF FAK1.alphaV_beta3_Integrin_Del1 IS NOT bound AND FAK1 IS NOT
7	IF FAK1.alphaV_beta3_Integrin_Del1 IS bound AND FAK1 IS NOT activ
8	IF FAK1.alphaV_beta3_Integrin_Del1 IS bound AND FAK1.Y0 IS NOT p
9	IF FAK1 IS active AND FAK1.Y0 IS phosphorylated AND FAK1.alphaV_I

Four tables are displayed, corresponding to compartments, components, reactions and the narrative itself. The user can edit each element by double-clicking on it. The editing is limited to specifying reaction rates, volumes and so on; PIDconvert does not aim to be a full-fledged narrative language development tool.

Once the user is satisfied, he can have the tool generate the narrative language program.

Conclusions

As we have seen, there are several challenges to the automated generation of narrative language models from the Pathway Interaction Database. Some of these can be met by further refining the PIDconvert tool; others will require some modification to the narrative language itself. Some can't be solved automatically, and in these cases the user will have to step in to direct the generation or modify the resulting program.

We will briefly review the major issues with the current version of PIDconvert:

- Inhibitors are not handled. Further work on PIDconvert could add support for inhibitors by treating them in the same way agents are treated, and considering them as a negative condition.
- Complexes are treated as single molecules. In theory, it should be possible to manage complexes within PIDconvert by adding binding sites for each other to the molecules that comprise the complex. However, this would lead to a quick proliferation of events and conditions; an event which involves a complex would have to be conditioned to each molecule of the complex being bound to each other. Further, this would exacerbate the problem of not knowing to which input an agent binds; not only we would have to make an assumption on which input to use, but also on which part of an input complex. A change to the narrative language syntax to explicitly address complexes would be preferable.
- Non-binary activation states aren't handled. This could be addressed within PIDconvert by adding multiple binary states, and arranging conditions in such a way that only one of these states can be true at the same time. Alternatively, non-binary states could be added to the narrative language.

  Either way, a potential problem could be caused by the tendency of PID to not specify activation states – for example, in reactions where a molecule is activated, typically the input is not specifically marked as "inactive". Currently, we assume that if the state marker is unspecified in the input and specified in the output, then the input has the opposite state than the output. However, if states are not binary, this assumption could no longer be viable.

- The proliferation of reactions can make it hard for the user to decide where to put reaction rates. We need to find a way to determine which sub-steps should happen immediately once conditions are met (and thus have infinite rate), and which should take a non-zero time instead. This could allow us to guide the user in the assignment of reaction rates.

Overall, there are still significant challenges to be overcome before we can confidently say that a majority of the PID can be automatically turned into a narrative language model, ready for the beta binders simulator. Still, this initial work has proven that the task is possible, and the potential reward – thousands of modeled interactions – should be easily worth it.

Bibliography

[1] Maria Luisa Guerriero, *From Intuitive Descriptions of Biochemical Systems to Their Formal Analysis*, PhD dissertation, DIT – University of Trento, December 2007.

[2] Centre for Computational and Systems Biology. http://www.cosbi.eu

[3] A. Romanel, L Dematté, C. Priami, *The Beta Workbench*, Centre for Computational and Systems Biology technical report TR-03-2007.

[4] The Pathway Interaction Database. http://pid.nci.nih.gov/

[5] BioCarta. http://www.biocarta.com

[6] Reactome. http://reactome.org