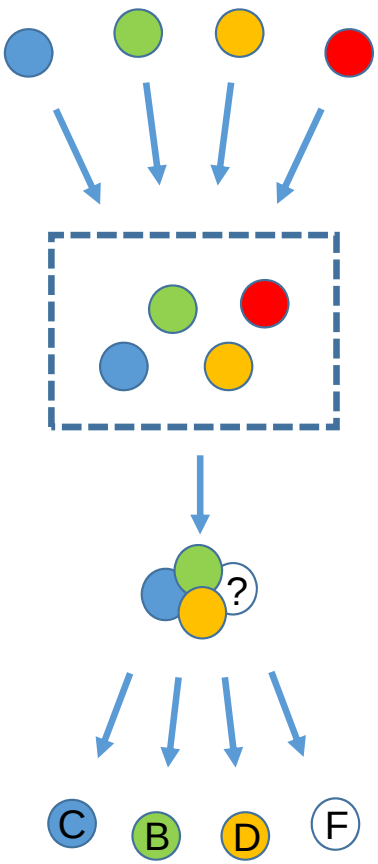
Protein-Protein Interactions

Introduction to Interaction Data Processing

February 2020

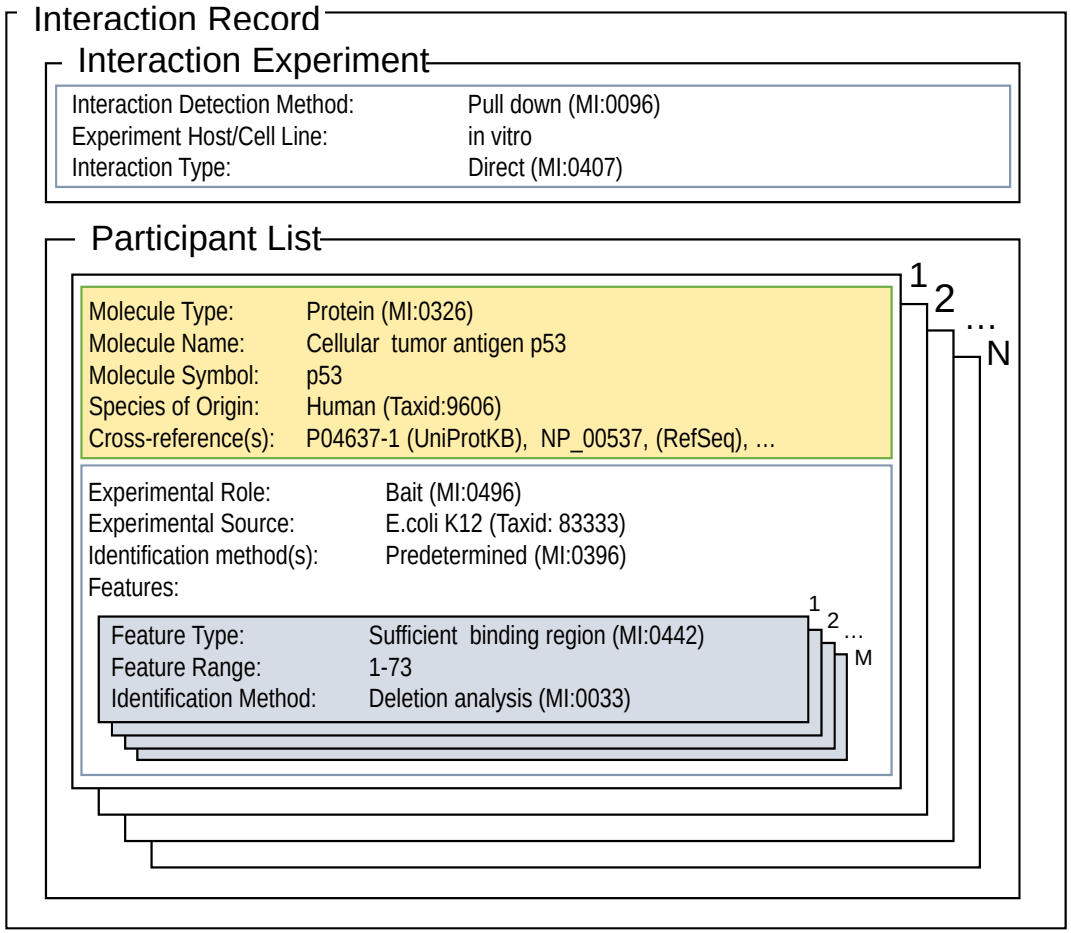
Lukasz Salwinski
lukasz@mbi.ucla.edu
Boyer Hall 205

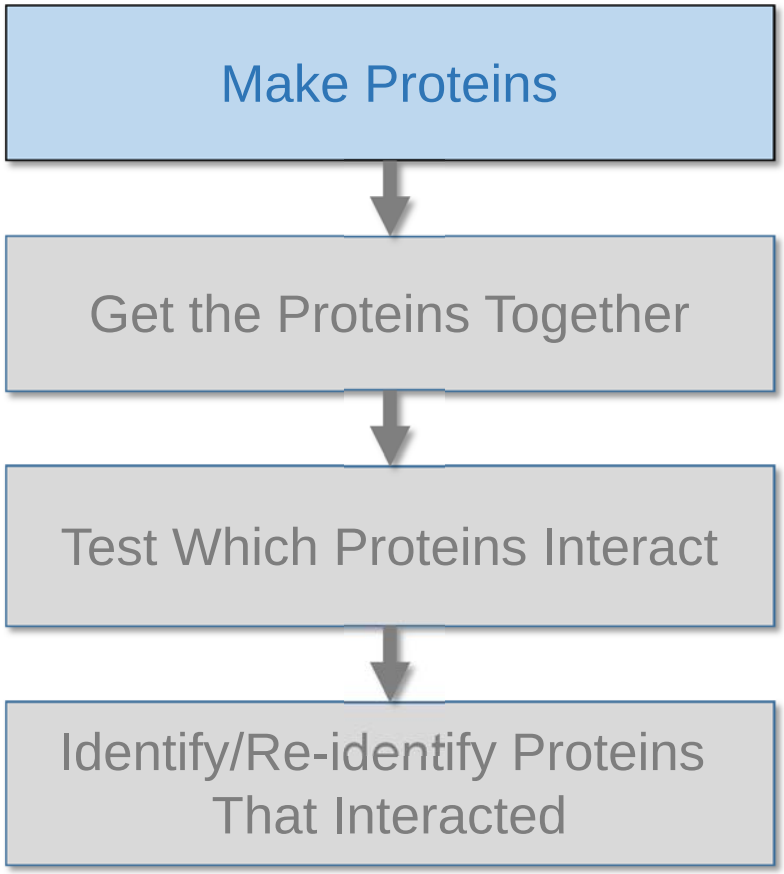
Interaction Experiment Flow



- Make proteins
  - Where: native vs heterologous host vs in vitro translation vs chemical synthesis
  - How much: native level vs overexpressed
  - Modifications: isoforms, fragments, mutations, PTMs present/absent
- Get them together
  - Where: native organism/cell type/tissue/compartment vs something else
  - When: cell cycle phase/cell state
- Test which ones interact
  - Diverse methods can be used to determine that that proteins interact
  - Information that can be inferred from each experiment depends on the method and experimental setup
- Identify proteins that interact
  - Identity of some proteins might be known a priori (eg purified, cloned/ tagged bait, etc)
  - Identity and/or state of some proteins might be ambiguous (eg unknown splice form, PTMs)
  - Some molecules participating in the interaction might remain unidentified

Interaction Experiment Record



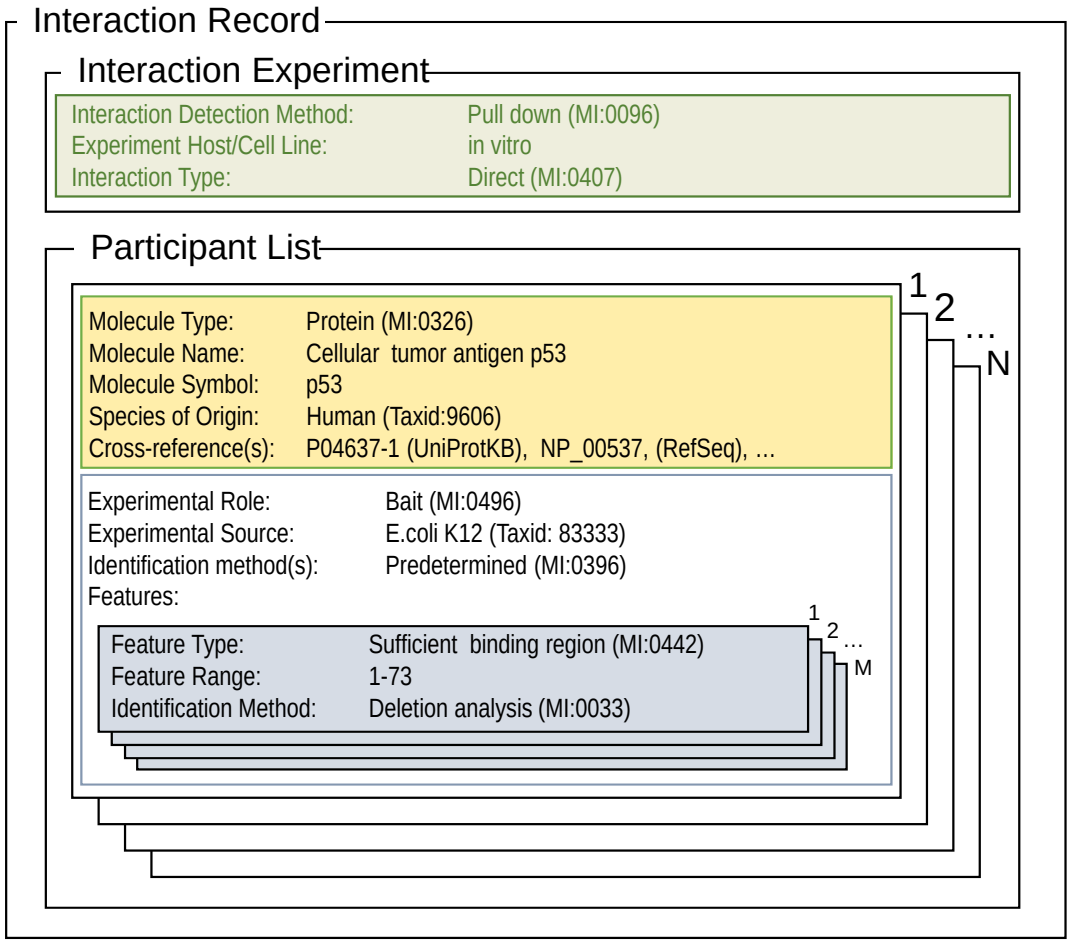


Interaction Database Records

Formats

Interaction Record Formats

PSI-MI XML (MIF) format



MIF/MI Definitions

HUPO HUMAN PROTEOME ORGANIZATION | translating the code of life | JOIN HUPO

Home > About HUPO

ABOUT HUPO

The Human Proteome Organization (HUPO) is an international scientific organization representing and promoting proteomics through international cooperation and collaborations by fostering the development of new technologies, techniques and training.

HUPO MISSION STATEMENT

To define and promote proteomics through international cooperation and collaborations by fostering the development of new technologies, techniques and training to better understand human disease.

Objectives

- Foster global collaboration in major proteomics projects by gathering leading international laboratories in life sciences, bioinformatics, mass spectrometry, systems biology, pathology, and medicine;
- Become the point of contact for proteomics research and commercialization activities worldwide;
- Support large-scale proteomics projects that are aimed at:
  - A mechanistic understanding of fundamental biological processes (often using model

HUPO HUMAN PROTEOME ORGANIZATION | translating the code of life | JOIN HUPO

Home > Initiatives > Proteomics Standards Initiative

PROTEOMICS STANDARDS INITIATIVE

Proteomics Standards Initiative

Website:

http://www.psidev.info/

Overview of Project

The HUPO Proteomics Standards Initiative (PSI) defines community standards for data representation in proteomics to facilitate data comparison, exchange and verification.

PSI Governance

Andy Jones, Chair

Eric Deutsch, Co-chair

Sandra Orchard, Co-chair

The main organizational unit of the Proteomics Standards Initiative is the work group. Currently, there are the following work groups:

- CompMS
- Early Career Researcher (ECR) Initiative
- Human Antibody Initiative
- Human Proteome Project
- Initiative on MultiOrganism Proteomes
- Pathology Pillar
- Proteomics Standards Initiative
- Clinical Proteome Tumor Analysis Consortium (CPTAC)

Enter search string

Interaction Record Formats

PSI-MI XML (MIF) format

Good

- Stable
  - o MIF 2.5 (2007)
  - o MIF 3.0 (2018; mostly backward-compatible)
- Database-neutral
  - o Developed by HUPO-PSI
- Widely used by data providers
  - o IMEx Consortium (DIP, IntAct, MINT,...) – native
  - o BioGRID – export
- Expressive enough to describe most of the interaction experiments
  - o Multi-protein interactions
  - o Protein features (PTMs, mutations, ...)
  - o Multiple experimental methods/protein
- Not limited to proteins
  - o Nucleic Acids
  - o Small Molecules

Bad

- Overly verbose
  - o Redundant open/close tags
  - o Several levels of nested elements
  - o But compresses quite well – 20x is not that rare
- Does not fit 'Excel spreadsheet' paradigm
  - o Not too surprising – interaction data is NOT tabular
- Limited set of good quality user-side tools
  - o Java JAMI is very versatile but complicated
- Limited support for reporting experiment ambiguities
  - o MIF 3.0 provides some support but curation lags
- No support for oligo-/poly-saccharides
  - o Limited by the current state of nomenclature
  - o No active curation (to my knowledge)
- XML is considered to be hard to work with

Interaction Record Formats

PSI-MI tab-delimited (MITAB) format

#ID(s) interactor A	ID(s) interactor B	Alt. ID(s) interactor A	Alt. ID(s) interactor B	Alias(es) interactor A	Alias(es) interactor B	Interaction detection method(s	Publicatio	Publication Identifier(s)	Taxid interactor A	Taxid inte	Interactio	Source da
uniprotkb:Q99728-1	uniprotkb:Q13526	intact:EBI-21498323	intact:EBI-714158|unip	psi-mi:q99728-1(display	psi-mi:pin1_human(disp	psi-mi:"MI:0007"(anti tag coimn	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:Q99728-1	uniprotkb:P38398	intact:EBI-21498323	intact:EBI-349905|unip	psi-mi:q99728-1(display	psi-mi:brca1_human(dis	psi-mi:"MI:0007"(anti tag coimn	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:P38398-1	uniprotkb:Q13526	intact:EBI-21498346	intact:EBI-714158|unip	psi-mi:p38398-1(display	psi-mi:pin1_human(disp	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:P38398-1	uniprotkb:Q06609-1	intact:EBI-21498346	intact:EBI-15557721	psi-mi:p38398-1(display	psi-mi:q06609-1(display	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:P38398-1	uniprotkb:Q99728	intact:EBI-21498346	intact:EBI-473181|unip	psi-mi:p38398-1(display	psi-mi:bard1_human(dis	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:P38398-1	uniprotkb:Q13526	intact:EBI-21498346	intact:EBI-714158|unip	psi-mi:p38398-1(display	psi-mi:pin1_human(disp	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:P38398	uniprotkb:Q13526	intact:EBI-349905|unip	intact:EBI-714158|unip	psi-mi:brca1_human(di	psi-mi:pin1_human(disp	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:Q99728-1	uniprotkb:Q13526	intact:EBI-21498323	intact:EBI-714158|unip	psi-mi:q99728-1(display	psi-mi:pin1_human(disp	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:Q99728-1	uniprotkb:Q13526	intact:EBI-21498323	intact:EBI-714158|unip	psi-mi:q99728-1(display	psi-mi:pin1_human(disp	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:Q13526	uniprotkb:Q99728-1	intact:EBI-714158|unip	intact:EBI-21498323	psi-mi:pin1_human(dis	psi-mi:q99728-1(display	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:P38398-1	uniprotkb:Q13526	intact:EBI-21498346	intact:EBI-714158|unip	psi-mi:p38398-1(display	psi-mi:pin1_human(disp	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
uniprotkb:P38398-1	uniprotkb:Q99728-1	intact:EBI-21498346	intact:EBI-21498323	psi-mi:p38398-1(display	psi-mi:q99728-1(display	psi-mi:"MI:0096"(pull down)	Daza-Mart	pubmed:31270457|imex:IN	taxid:9606(human)|	taxi	psi-mi:"M	psi-mi:"M i
u			intact:			"MI:0096"(pull down)	Daza-Mart	pubmed:3127				M i
u			intact:			"MI:0096"(pull down)	Daza-Mart	pubmed:3127				M i
u			intact:			"MI:0096"(pull down)	Daza-Mart	pubmed:3127				M i
u			intact:			"MI:0096"(pull down)	Daza-Mart	pubmed:3127				M i
u		ip	intact:			"MI:0007"(anti tag coimn	Daza-Mart	pubmed:3127				M i

MITAB 2.5

- ID(s) interactor A
- ID(s) interactor B
- Alt. ID(s) interactor A
- Alt. ID(s) interactor B
- Alias(es) interactor A
- Alias(es) interactor B
- Interaction detection method(s)
- Publication 1st author(s)
- Publication Identifier(s)
- Taxid interactor A
- Taxid interactor B
- Interaction type(s)
- Source database(s)
- Interaction identifier(s)
- Confidence value(s)

MITAB 2.6

- Expansion method(s)
- Biological role(s) interactor A
- Biological role(s) interactor B
- Experimental role(s) interactor A
- Experimental role(s) interactor B
- Type(s) interactor A
- Type(s) interactor B
- Xref(s) interactor A
- Xref(s) interactor B
- Interaction Xref(s)

MITA

- Annotation(s) interactor A
- Annotation(s) interactor B
- Interaction annotation(s)
- Host organism(s)
- Interaction parameter(s)
- Creation date
- Update date
- Checksum(s) interactor A
- Checksum(s) interactor B
- Interaction Checksum(s)
- Negative

MITAB 2.7

- Feature(s) interactor A
- Feature(s) interactor B
- Stoichiometry(s) interactor A
- Stoichiometry(s) interactor B
- Identification method participant A
- Identification method participant B

Column values

XREF	VALUE	DESCRIPTION
uniprotkb:	P12345	(very important protein)
wwpdb:	1COL|wwpdb:3COL	
psi-mi:	"MI:0096"	(pull-down)
psi-mi:	"MI:0097"	("\"super\" tag")

Interaction Record Formats

PSI-MI tab-delimited (MITAB) format

Good

- Easy to read into a spreadsheet
- Supported by third-party libraries

Bad

- Applicable only to binary interactions
  - Cannot handle multi-protein complexes
- Many columns can be multi-valued
  - Requires custom parsing routines
- Some information originally available in MIF format cannot be stored as MITAB
  - The format is lossy – is, essentially, impossible, to restore fully-featured MIF record from its MITAB representation
- Less stable than MIF
  - MITAB 2.5, 2.7, 2.8 (MIF 2.5 derivatives)
  - MITAB 3.0 (3.0 derivative)

Interaction Record Formats

BioGRID tab & complex tab formats

Good

- Easy to read into a spreadsheet

Bad

- Separate format for binary and multi-protein complexes
- Supports only protein-protein interactions
- Identifies proteins by gene identifies
  - It works only for organisms that do not splice
- Provides less information than PSI-MI MIF
  - This is because BioGRID extracts less information about interactions that IMEx Consortium databases
- Uses simplified, non-standard CV terms
- See also MITAB deficiencies

Interaction Record Formats

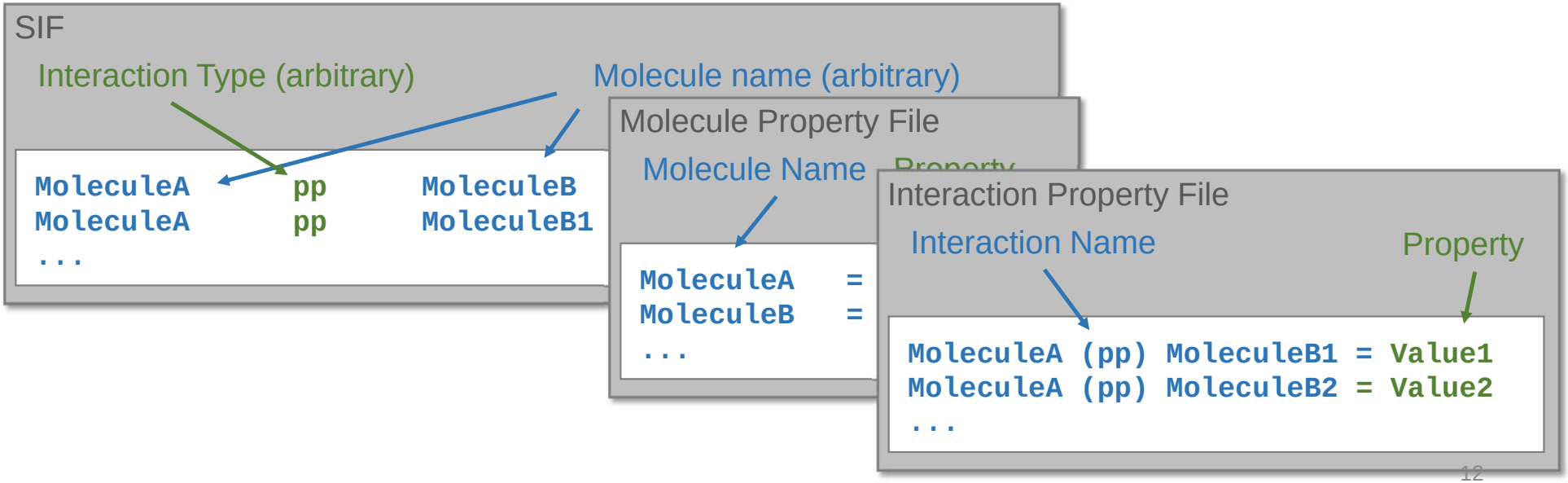
Cytoscape SIF format

Good

- Native Cytoscape format
- Simple/easy to prepare (spreadsheet will work)
- Not limited to biological data

Bad

- Only binary interactions
- Must be combined with information from additional files in order to provide more detailed information



Interaction Data Processing

Topics

- XML parsing
  - Lxml library
  - Xpath
  - Parsing UniprotKB XML files
- MIF file structure
  - ‘compact vs ‘expanded’ variants
- Binary expansion of multi-protein interactions
  - ‘spoke’ vs ‘matrix’ expansion
- Extracting data from MIF files
  - Access relevant/useful information
  - Prepare files ready to import into Cytoscape

XML File Anatomy

```
<?xml version="1.0" encoding="UTF-8"?>
<mif:entrySet xmlns:mif="http://psi.hupo.org/mi/mif"
              level="2" version="5" minorVersion="4">
 <mif:entry>
  <mif:source releaseDate="2019-05-13">
   <mif:names>
     <mif:shortLabel>DIP</mif:shortLabel>
     <mif:fullName>Database of Interacting Proteins</mif:fullName>
   </mif:names>
   <mif:xref>
      <mif:primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0465"
                      refType="identity" refTypeAc="MI:0356"/>
      ...
   </mif:xref>
  </mif:source>
   ...
 </mif:entry>
</mif:entrySet>
```

Namespace

Text

Element

Opening tag

Closing tag

Attribute (name = “value”)

Namespace Prefix

<ROOT ELEMENT>...</ROOT ELEMENT>

`<element attribute=“value”/>`

is equivalent to

`<element attribute=“value”></element>`

XML File Parsing

Lxml library (https://lxml.de/)

```
from lxml import etree
from io import StringIO
```

Only if needed...

```
xml = '<protein acc="P60010"><seq>MKYDDEW...</seq></protein>'
```

```
strDom = etree.fromstring( xml )
```

or

```
strDom = etree.fromstring( StringIO(xml) )
```

Parse String

```
fileDom = etree.parse("doc/test.xml")
```

or

```
fileDom = etree.parse( open("doc/test.xml") )
```

Parse File/URL

.gz files OK

```
print( etree.tostring( strDom ).decode())
```

... and back to String

See lxml web site for more options

XML File Parsing

lxml library (https://lxml.de/)

```
from lxml import etree

xml = '<protein acc="P60010"><seq format="fasta">MKYDDEW...</seq></protein>'

xmlDom = etree.fromstring( xml )

for child in xmlDom:
```

`print( child.tag.decode() )`	Get element tag
`print( child.get("format").decode() )`	Get attribute
`print( child.text.decode() )`	Get text
`print( etree.tostring(child).decode() )`	Get element as XML

See lxml web site for more options

XML File Parsing

lxml Xpath support (https://lxml.de/xpathxslt.html)

```
from lxml import etree

xml = '<protein acc="P60010"><seq>MKYDDEW...</seq></protein>'

xmlDom = etree.fromstring( xml )

root = xmlDom.xpath('/protein')
```

Get top-level ‘protein’ element

NOTE: Returns a list !!!

```
for child in root[0]:

    print( child.tag.decode() )
```

Get element tag

```
    print( child.get("format").decode() )
```

Get attribute

```
    print( etree.tostring(child).decode() )
```

Get element as XML

See lxml web site for more options

XML File Parsing

lxml Xpath support (https://lxml.de/xpathxslt.html)

```
from lxml import etree

xml = '<protein acc="P60010"><seq>MKYDDEW...</seq></protein>'

xmlDom = etree.fromstring( xml )
```

```
t1 = xmlDom.xpath('/protein/seq/text()')
```

Get the text of 'seq' elements that are children of the top-level 'protein' element

```
t2 = xmlDom.xpath('//seq/text()')
```

Get the text of ANY 'seq' element

```
e3 = xmlDom.xpath('//protein[@acc="P60010"]/seq')
```

LIST!!!

Get 'seq' elements that are children of 'protein' element with 'acc' attribute equals to 'P60010'

```
e4 = e3[0].xpath('./text()')
```

Get the text of the current element

See *https://en.wikipedia.org/wiki/XPath* and *https://www.w3.org/TR/xpath-10* for more details

XML File Parsing

lxml Xpath namespace support (https://lxml.de/xpathxslt.html)

```
from lxml import etree

xml = '''<mif:protein xmlns:mif="http://psi.hupo.org/mi/mif" acc="P60010">
          <mif:seq>MKYDDEW...</mif:seq>
       </mif:protein>'''

xmlDom = etree.fromstring( xml )

e = xmlDom.xpath('/m:protein/m:seq',
                 namespaces={'m': 'http://psi.hupo.org/mi/mif'})
```

```
print( e[0].tag.decode())
```

Get qualified (i.e. with namespace) tag

```
qname = etree.QName(e[0])

print( qname.localname.decode() )
print( qname.namespace.decode() )
```

Split qualified tag into namespace and local name

XML File Parsing

Example: Parsing UniprotKB records

https://www.uniprot.org/uniprot/P60010.xml


```xml
<?xml version='1.0' encoding='UTF-8'?>
<uniprot xmlns="http://uniprot.org/uniprot" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://u
<entry dataset="Swiss-Prot" created="1986-07-21" modified="2019-12-11" version="177">
<accession>P60010</accession>
<accession>D6VTJ1</accession>
<accession>P02579</accession>
<accession>Q9P3X6</accession>
<accession>Q9P3X7</accession>
<name>ACT_YEAST</name>
<protein>
<recommendedName>
<fullName>Actin</fullName>
</recommendedName>
</protein>
<gene>
<name type="primary">ACT1</name>
<name type="synonym">ABY1</name>
<name type="synonym">END7</name>
<name type="ordered locus">YFL039C</name>
</gene>
<organism>
<name type="scientific">Saccharomyces cerevisiae (strain ATCC 204508 / S288c)</name>
<name type="common">Baker's yeast</name>
<dbReference type="NCBI Taxonomy" id="559292"/>
<lineage>
<taxon>Eukaryota</taxon>
<taxon>Fungi</taxon>
<taxon>Dikarya</taxon>
<taxon>Ascomycota</taxon>
<taxon>Saccharomycotina</taxon>
<taxon>Saccharomycetes</taxon>
<taxon>Saccharomycetales</taxon>
<taxon>Saccharomycetaceae</taxon>
<taxon>Saccharomyces</taxon>
</lineage>
</organism>
<reference key="1">
<citation type="journal article" date="1980" name="Proc. Natl. Acad. Sci. U.S.A." volume="77" first="2546" last="2550">
<title>Structure of a split yeast gene: complete nucleotide sequence of the actin gene in Saccharomyces cerevisiae.</title>
<authorList>
<person name="Gallwitz D."/>
<person name="Sures I."/>
</authorList>
<dbReference type="PubMed" id="6994099"/>
<dbReference type="DOI" id="10.1073/pnas.77.5.2546"/>
</citation>
<scope>NUCLEOTIDE SEQUENCE [GENOMIC DNA]</scope>
</reference>
<reference key="2">
<citation type="journal article" date="1980" name="Proc. Natl. Acad. Sci. U.S.A." volume="77" first="3912" last="3916">
<title>Isolation and sequence of the gene for actin in Saccharomyces cerevisiae.</title>
<authorList>
<person name="Ng R."/>
<person name="Abelson J."/>
```


Lots and lots of stuff !!!

- Primary/secondary (aka past) accessions
- Names
  - ❑ Protein Names (including aliases)
  - ❑ Gene Names (including aliases)
- Taxonomy
  - ❑ Species of origin (possibly strain)
  - ❑ Linage
- Sequence (possibly isoforms)
  - ❑ Molecular weight
- Cross-references
  - ❑ Articles
  - ❑ Gene Ontology Terms
  - ❑ Domains (InterPro, Pfam Prosite)
  - ❑ Structures (PDB)
  - ❑ Interactions
- And more....

XML File Parsing

Example: Parsing UniprotKB records

uniprotPrint.py

```python
#!/usr/bin/python3

import sys
from lxml import etree
from urllib.request import urlopen

uniprotUrl = "https://www.uniprot.org/uniprot/%%ACC%%.xml"

assert len(sys.argv) == 2
accession = sys.argv[1]

accessionUrl = uniprotUrl.replace("%%ACC%%",accession)

xmlDom = etree.parse( urlopen( accessionUrl ))
print( etree.tostring( xmlDom ).decode() )
```

Needed for https only

Program flow

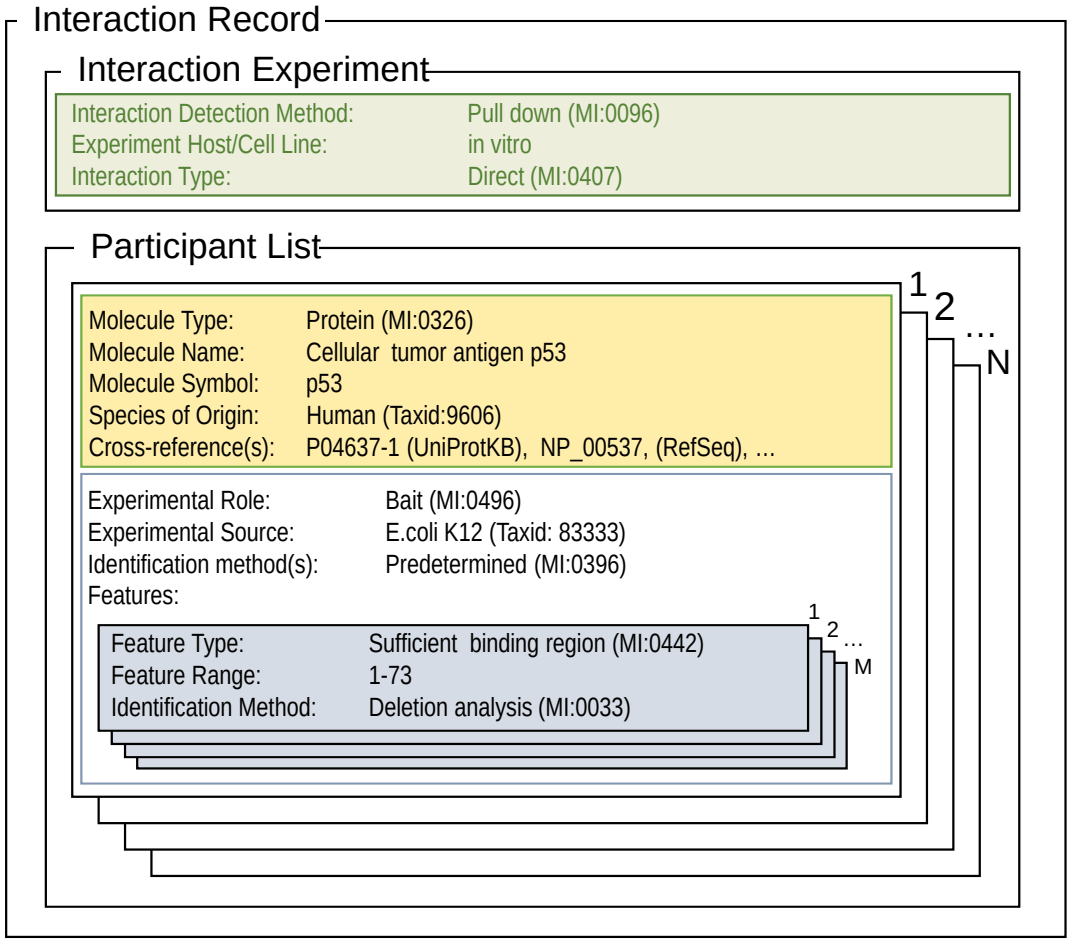
- Specify UniprotKB Identifier
- Read Corresponding File
- Find interesting element(s)
- Print results

uniprotRead.py

```python
#!/usr/bin/python3

import sys
from lxml import etree
from urllib.request import urlopen

uprotUrl = "https://www.uniprot.org/uniprot/%%ACC%%.xml"
uprotNs = { 'up': 'http://uniprot.org/uniprot'}
assert len(sys.argv) == 2
accession = sys.argv[1]

accessionUrl = uprotUrl.replace("%%ACC%%",accession)
uprotTree = etree.parse( urlopen( accessionUrl ))

# find name
uprotNameList = uprotTree.xpath( "//up:protein//up:fullName/text()",
                                 namespaces=uprotNs)

for name in uprotNameList:
    print( "Name: %s" % ( name))

# find accessions
uprotAccList = uprotTree.xpath( "//up:accession/text()",
                                namespaces=uprotNs)
accType = "primary"
for acc in uprotAccList:
    print("Accession: %s (%s)" % ( acc, accType ))
    accType ="secondary"


# find gene names
uprotGeneNameList = uprotTree.xpath( "//up:gene/up:name",
                                     namespaces=uprotNs)
for gname in uprotGeneNameList:
    gnType = gname.xpath( "./@type" )
    gnName = gname.xpath( "./text()" )
    print("Gene Name: %s  (%s)" % (gnName[0] , gnType[0] ))

# find sequence
uprotSequenceList = uprotTree.xpath( "//up:sequence",
                                     namespaces=uprotNs)
for seq in uprotSequenceList:
    seqMass = seq.xpath( "./@mass")
    seqVersion = seq.xpath( "./@version")
    seqStr = seq.xpath( "./text()")

    print( "Sequence (ver: %s): %s " % ( seqVersion[0], seqStr[0] ))
    print( "Mass: %6.1fkD" %(int(seqMass[0])/1000))
```

Interaction Record Formats

PSI-MI XML (MIF) format



Interaction Data Processing

PSI-MI XML (MIF) file anatomy

Compact MIFs use references to interactors and/or experiments

```
<entry Set>
  <entry>
    <source releaseDate='2019-05-22'>...</source>
    <interactorList>
      <interactor id='1'>...</interactor>
      <interactor id='2'>...</interactor>
      ...
    </interactorList>
    <experimentList>
      <experiment id='1'>...</experiment>
      <experiment id='2'>...</experiment>
      ...
    </experimentList>
    <interactionList>
      <interaction id='1' imexId='IM-123456-1'>
        <experimentList>
          <experimentRef id='1'/>
        </experimentList>
        <participantList>
          <participant id='1'>
            <interactorRef id='2'/>
            <biologicalRole>...</biologicalRole>
            <experiemntalRole>...</experimentalRole>
            ....
          </participant>
          <participant id='2'>...</participant>
          ....
        </participantList>
      </interaction>
      <interaction id='2' imexId='IM-123456-12'>...</interaction>
        ...
    </interactionList>
  </entry>
  ...
</entrySet>
```

Record source (database)

List of interactors

List of experimental methods/protocols

List of interactions

Reference to experimental method/protocol

Reference to interactor

Participant (e.g. protein)

- Describes the state of the molecule as used in the experiment
- Refers to 'interactor' – the reference description of the molecule (e.g. UniprotKB)

Interaction Data Processing

PSI-MI XML (MIF) file anatomy

```
<entry Set>
  <entry>
    <source releaseDate='2019-05-22'>...</source>
    <interactionList>
      <interaction id='1' imexId='IM-123456-1'>
        <experimentList>
          <experiment id='1'/>...</experiment>
        </experimentList>
        <participantList>
          <participant id='1'>
            <interactor id='2'>...</interactor>
            <biologicalRole>...</biologicalRole>
            <experiemntalRole>...</experimentalRole>
            ....
          </participant>
          <participant id='2'>...</participant>
          ....
        </participantList>
      </interaction>
      <interaction id='2' imexId='IM-123456-12'>...</interaction>
       ...
    </interactionList>
  </entry>
  ...
</entrySet>
```

Record source (database)

List of interactions

Experimental method/protocol description

Interactor description

Expanded MIFs describe interactors and experiments within each interaction

Conversion between Compact & Expanded MIF is lossless !!!

Participant (e.g. protein)

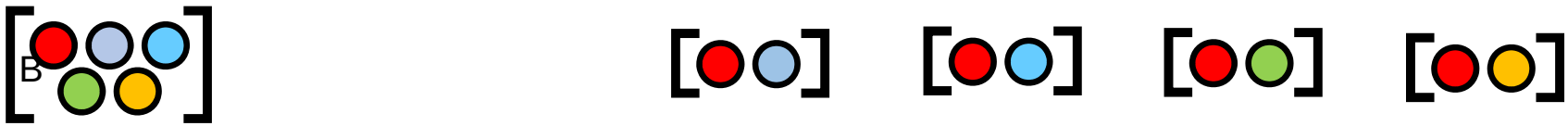
- Describes the state of the molecule as used in the experiment
- Refers to 'interactor' – the reference description of the molecule (e.g. UniprotKB)

Interaction Data Processing

Binary Expansion

Binary Expansion is is lossy !!!

Association → Physical Associations



Physical Association → Physical Associations

Direct → Direct

Interaction Data Processing

PSI-MI XML (MIF) file anatomy

Compact MIFs use references to interactors and/or experiments

Description of the experimental method (explicit or a reference)

Description of the interactor (explicit or a reference)

Participant (e.g. protein)

- Describes the state of the molecule as used in the experiment
- Refers to 'interactor' – the reference description of the molecule (e.g. UniprotKB)

Features (e.g. mutations)

- Modifications of the molecule relative to the reference state

Interaction Data Processing

Data structures

Protein (reference state/interactor)

- ❑ Nested dictionary

Protein collection

- ❑ Dictionary, using unique id as key

Interaction evidence

- ❑ Nested dictionary
- ❑ Participants as list
  - ❑ Each refers to interactor through its unique id

Interaction evidence collection

- ❑ List

  or
- ❑ Dictionary, using IMEx id as key

```
prot1 = { "accession" : "P60010",
          "taxonId" : 4932,
          "seqStr" : "MDSEVAALVIDNGSGMCKAGFAGDD...",
          ... }



prot2 = { "accession": "P60010",
          "taxon" : { "taxId": 4932,
                      "sciName" : "S. cerevisiae",
                      "comName" : "budding yeast" },
          "seqStr": ['M','D','S',...],
          "GO": [{ "id" :"GO:0016573", value: "P:histone acetylation"},
                 { "id": "GO:0006887", value: "P:exocytosis" }].
          ... }



protList = [ {}, {}, {} ]

                or

protDict = { "P60010": prot2, ... }

                    Most often easier to work with


evid1 = { "imexId" : "IM-1234-1",
          "ppantList" : [{ "protId": "P60010",
                           "expHost": 83333,
                           "features": [feature1, feature2,....],
                           "idMethods": ["MI:0123",....],
                           ...},
                         { "protId": "P61234",
                           "expHost": 4932,
                           "features": [feature3, feature4, ...],
                           ...],
           "intType": {"id": "MI:0912", "value": "direct"},
           "detectionMth": {"id": "MI:0324", "value": "pull-down"},
           ... }


evidList = [ evid1, evid2, evid3, ... ]
```

MIF File Parsing

Preliminaries/Record Source Information

mifReadFile.py

```python
#!/usr/bin/python3

import sys
from lxml import etree

mifNs = { 'm': 'http://psi.hupo.org/mi/mif'}
assert len(sys.argv) == 2
mifFile = sys.argv[1]

mifTree = etree.parse( mifFile )

# find record source
mifSrcName = mifTree.xpath( "//m:source//m:shortLabel/text()",
                            namespaces=mifNs)
print( "Record Source: %s" % ( mifSrcName[0] ))

# find record release date
mifDate = mifTree.xpath( "//m:source/@releaseDate",
                         namespaces=mifNs)
print("Release Date: %s" % ( mifDate[0] ))
```

Program flow

- Specify input file location
- Read and parse the input file
- Find interesting element(s)
- Print out/save results

```
1     <?xml version="1.0" encoding="UTF-8"?>
2     <entrySet xmlns="http://psi.hupo.org/mi/mif" xmlns:xsi="http://www.w3.o
3       <entry>
4         <source releaseDate="2019-05-13">
5           <names>
6             <shortLabel>DIP</shortLabel>
7             <fullName>Database of Interacting Proteins</fullName>
8           </names>
9           <xref>
10            <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0465" refType="id
11            <secondaryRef db="pubmed" dbAc="MI:0446" id="14681454" refType=
12            <secondaryRef db="intact" dbAc="MI:0469" id="EBI-1579232" refTy
13          </xref>
14          <attributeList>
15            <attribute name="postaladdress">611 Young Drive East; Los Angel
16            <attribute name="url" nameAc="MI:0614">http://dip.doe-mbi.ucla.
17            <attribute name="contact-email" nameAc="MI:0634">dip@mbi.ucla.e
18          </attributeList>
19        </source>
20        <experimentList>
21          <experimentDescription id="1">
76          <experimentDescription id="2">
133       </experimentList>
134       <interactorList>
135         <interactor id="3">
206         <interactor id="4">
266         <interactor id="5">
327         <interactor id="6">
409         <interactor id="7">
466         <interactor id="8">
535         <interactor id="9">
595         <interactor id="10">
650         <interactor id="11">
701         <interactor id="12">
758         <interactor id="13">
832         <interactor id="14">
916         <interactor id="15">
967         <interactor id="16">
1022      </interactorList>
1023      <interactionList>
1024        <interaction id="17">
1843        <interaction id="33">
2501      </interactionList>
2502    </entry>
2503  </entrySet>
```

MIF File Parsing

Experimental Method Information

```python
exByRefId = {}

#find experiments
mifExList = mifTree.xpath( "//m:experimentDescription",
                           namespaces=mifNs)
for ex in mifExList:

    experiment = {}

    # experiment id attribute: unique *within individual file*
    exId = ex.xpath( "./@id",
                     namespaces=mifNs)
    exByRefId[ exId[0] ] = experiment

    #pubmed
    exPmid = ex.xpath( "./m:bibref/m:xref/*[@db='pubmed']/@id",
                       namespaces=mifNs)
    if exPmid:
        experiment['pmid'] = exPmid[0]

    #experiment host
    exHost = ex.xpath( "./m:hostOrganismList/m:hostOrganism",
                       namespaces=mifNs)
    hostList = []
    experiment['hostList'] = hostList
    for host in exHost:
        curHost = {}
        hostList.append( curHost )

        hostName = host.xpath( "./m:names/m:shortLabel/text()",
                               namespaces=mifNs )
        hostTaxid = host.xpath( "./@ncbiTaxId",
                                namespaces=mifNs )

        curHost['name'] = hostName[0]
        curHost['taxid'] = hostTaxid[0]

```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<entrySet xmlns="http://psi.hupo.org/mi/mif" xmlns:xsi="http://www.w3.
  <entry>
    <source releaseDate="2019-05-13">
    <experimentList>
      <experimentDescription id="1">
        <names>
          <fullName>High-resolution cryo-EM analysis of the yeast ATP
        </names>
        <bibref>
          <xref>
            <primaryRef db="pubmed" dbAc="MI:0446" id="29650704" refTy
            <secondaryRef db="intact" dbAc="MI:0469" id="EBI-20595933"
          </xref>
        </bibref>
        <xref>
          <primaryRef db="pubmed" dbAc="MI:0446" id="29650704" refType
        </xref>
        <hostOrganismList>
          <hostOrganism ncbiTaxId="559292">
            <names>
              <shortLabel>yeast</shortLabel>
              <fullName>Saccharomyces cerevisiae</fullName>
              <alias type="synonym" typeAc="MI:1041">Baker's yeast</al
            </names>
          </hostOrganism>
        </hostOrganismList>
        <interactionDetectionMethod>
          <names>
            <shortLabel>pull down</shortLabel>
            <fullName>pull down</fullName>
          </names>
          <xref>
            <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0096" refTyp
            <secondaryRef db="intact" dbAc="MI:0469" id="EBI-1223" ref
            <secondaryRef db="pubmed" dbAc="MI:0446" id="14755292" ref
          </xref>
        </interactionDetectionMethod>
        <participantIdentificationMethod>
          <names>
            <shortLabel>weight identificat</shortLabel>
            <fullName>confirmation by molecular weight</fullName>
            <alias type="synonym" typeAc="MI:1041">weight identificat<
          </names>
          <xref>
```

MIF File Parsing

Experimental Method Information

```python
    exByRefId[ exId[0] ] = experiment

    #pubmed
    exPmid = ex.xpath( "./m:bibref/m:xref/*[@db='pubmed']/@id",
                       namespaces=mifNs)
    if exPmid:
        experiment['pmid'] = exPmid[0]

    #experiment host
    exHost = ex.xpath( "./m:hostOrganismList/m:hostOrganism",
                       namespaces=mifNs)
    hostList = []
    experiment['hostList'] = hostList
    for host in exHost:
        curHost = {}
        hostList.append( curHost )

        hostName = host.xpath( "./m:names/m:shortLabel/text()",
                               namespaces=mifNs )
        hostTaxid = host.xpath( "./@ncbiTaxId",
                                namespaces=mifNs )

        curHost['name'] = hostName[0]
        curHost['taxid'] = hostTaxid[0]

    # detection method
    exDetMth = ex.xpath( "./m:interactionDetectionMethod",
                         namespaces=mifNs)
    detMthName = exDetMth[0].xpath( "./m:names/m:shortLabel/text()",
                                    namespaces=mifNs )
    detMthId = exDetMth[0].xpath( "./m:xref/*[@db='psi-mi']/@id",
                                  namespaces=mifNs )

    experiment['detMeth'] = { 'name': detMthName[0],
                              'cv': 'psi-mi',
                              'id': detMthId[0] }
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<entrySet xmlns="http://psi.hupo.org/mi/mif" xmlns:xsi="http://www.w3.
  <entry>
    <source releaseDate="2019-05-13">
    <experimentList>
      <experimentDescription id="1">
        <names>
          <fullName>High-resolution cryo-EM analysis of the yeast ATP
        </names>
        <bibref>
          <xref>
            <primaryRef db="pubmed" dbAc="MI:0446" id="29650704" refTy
            <secondaryRef db="intact" dbAc="MI:0469" id="EBI-20595933"
          </xref>
        </bibref>
        <xref>
          <primaryRef db="pubmed" dbAc="MI:0446" id="29650704" refType
        </xref>
        <hostOrganismList>
          <hostOrganism ncbiTaxId="559292">
            <names>
              <shortLabel>yeast</shortLabel>
              <fullName>Saccharomyces cerevisiae</fullName>
              <alias type="synonym" typeAc="MI:1041">Baker's yeast</al
            </names>
          </hostOrganism>
        </hostOrganismList>
        <interactionDetectionMethod>
          <names>
            <shortLabel>pull down</shortLabel>
            <fullName>pull down</fullName>
          </names>
          <xref>
            <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0096" refTy
            <secondaryRef db="intact" dbAc="MI:0469" id="EBI-1223" re
            <secondaryRef db="pubmed" dbAc="MI:0446" id="14755292" re
          </xref>
        </interactionDetectionMethod>
        <participantIdentificationMethod>
          <names>
            <shortLabel>weight identificat</shortLabel>
            <fullName>confirmation by molecular weight</fullName>
            <alias type="synonym" typeAc="MI:1041">weight identificat<
          </names>
          <xref>
```

MIF File Parsing

Interactor (Reference Molecule) Information (I)

```python
irByRefId = {}
irByAccId = {}

#find interactors
mifIrList = mifTree.xpath( "//m:interactor",
                           namespaces=mifNs)
for ir in mifIrList:

    interactor = {}

    # primary identifier: *shoud* be unique across *all files*
    primaryIdDB = ir.xpath( "./m:xref/m:primaryRef/@db",
                            namespaces=mifNs)
    primaryIdAC = ir.xpath( "./m:xref/m:primaryRef/@id",
                            namespaces=mifNs)

    interactor['acc'] = primaryIdDB[0]+":"+primaryIdAC[0]
    irByAccId[ interactor['acc'] ] = interactor

    # interactor id attribute: unique *only* within individual file
    irId = ir.xpath( "./@id",
                     namespaces=mifNs)
    irByRefId[ irId[0] ] = interactor

    # short label
    label = ir.xpath( "./m:names/m:shortLabel/text()",
                      namespaces=mifNs)
    if label:
        interactor['label'] = label[0]
    else:
        interactor['label'] =''

    # interactor (molecule) type
    irType = ir.xpath( "./m:interactorType",
                       namespaces=mifNs )
    irTypeName = irType[0].xpath( "./m:names/m:shortLabel/text()",
                                  namespaces=mifNs )
```

```
<experimentList>
  <experimentDescription id="1">
  <experimentDescription id="2">
</experimentList>
<interactorList>
  <interactor id="3">
    <names>
      <shortLabel>atpg_yeast</shortLabel>
      <fullName>ATP synthase subunit gamma, mitochondrial</fullNam
      <alias type="gene name" typeAc="MI:0301">ATP3</alias>
      <alias type="locus name" typeAc="MI:0305">YBR039W</alias>
      <alias type="orf name" typeAc="MI:0306">YBR0408</alias>
      <alias type="gene name synonym" typeAc="MI:0302">F-ATPase ga
      <alias type="gene name synonym" typeAc="MI:0302">ATP3a</alia
      <alias type="gene name synonym" typeAc="MI:0302">ATP3b</alia
    </names>
    <xref>
      <primaryRef db="uniprotkb" dbAc="MI:0486" id="P38077" versic
      <secondaryRef db="uniprotkb" dbAc="MI:0486" id="Q54AF5" vers
      <secondaryRef db="uniprotkb" dbAc="MI:0486" id="Q76MT6" vers
      <secondaryRef db="uniprotkb" dbAc="MI:0486" id="D6VQ39" vers
      <secondaryRef db="intact" dbAc="MI:0469" id="EBI-3271" refTy
      <secondaryRef db="go" dbAc="MI:0448" id="GO:0005756"/>
      <secondaryRef db="go" dbAc="MI:0448" id="GO:0046933"/>
      <secondaryRef db="go" dbAc="MI:0448" id="GO:0015986"/>
      <secondaryRef db="refseq" dbAc="MI:0481" id="NP_009595.1"/>
      <secondaryRef db="sgd" dbAc="MI:0484" id="S000000243"/>
      <secondaryRef db="interpro" dbAc="MI:0449" id="IPR000131"/>
      <secondaryRef db="interpro" dbAc="MI:0449" id="IPR023632"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="2HLD"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="2WPD"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="2XOK"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="3FKS"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="3OE7"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="3OEE"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="3OEH"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="3OFN"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="3ZRY"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="4B2Q"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="3ZIA"/>
      <secondaryRef db="dip" dbAc="MI:0465" id="DIP-3035N"/>
      <secondaryRef db="interpro" dbAc="MI:0449" id="IPR035968"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="6B8H"/>
      <secondaryRef db="mint" dbAc="MI:0471" id="P38077"/>
      <secondaryRef db="rcsb pdb" dbAc="MI:0460" id="6CP3"/>
```

MIF File Parsing

Interactor (Reference Molecule) Information (II)

```python
85      primaryIdAC = ir.xpath( "./m:xref/m:primaryRef/@id",
86                              namespaces=mifNs)
87
88      interactor['acc'] = primaryIdDB[0]+":"+primaryIdAC[0]
89      irByAccId[ interactor['acc'] ] = interactor
90
91      # interactor id attribute: unique *only* within individual file
92      irId = ir.xpath( "./@id",
93                       namespaces=mifNs)
94      irByRefId[ irId[0] ] = interactor
95
96      # short label
97      label = ir.xpath( "./m:names/m:shortLabel/text()",
98                        namespaces=mifNs)
99      if label:
100         interactor['label'] = label[0]
101     else:
102         interactor['label'] =''
103
104     # interactor (molecule) type
105     irType = ir.xpath( "./m:interactorType",
106                        namespaces=mifNs )
107     irTypeName = irType[0].xpath( "./m:names/m:shortLabel/text()",
108                                   namespaces=mifNs )
109     irTypeId = irType[0].xpath( "./m:xref/*[@db='psi-mi']/@id",
110                                 namespaces=mifNs )
111
112     interactor['type'] = { 'name': irTypeName[0],
113                            'cv': 'psi-mi',
114                            'id': irTypeId[0] }
115
116     # gene name
117     gene = ir.xpath( "./m:names/m:alias[@type='gene name']/text()",
118                      namespaces=mifNs)
119     if gene:
120         interactor['gene'] = gene[0]
121
```

```
134     <interactorList>
135       <interactor id="3">
136         <names>
137           <shortLabel>atpg_yeast</shortLabel>
138           <fullName>ATP synthase subunit gamma, mitochondrial</fullNam
139           <alias type="gene name" typeAc="MI:0301">ATP3</alias>
140           <alias type="locus name" typeAc="MI:0305">YBR039W</alias>
141           <alias type="orf name" typeAc="MI:0306">YBR0408</alias>
142           <alias type="gene name synonym" typeAc="MI:0302">F-ATPase ga
143           <alias type="gene name synonym" typeAc="MI:0302">ATP3a</alia
144           <alias type="gene name synonym" typeAc="MI:0302">ATP3b</alia
145         </names>
146         <xref>
181         <interactorType>
182           <names>
183             <shortLabel>protein</shortLabel>
184             <fullName>protein</fullName>
185           </names>
186           <xref>
187             <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0326" refTy
188             <secondaryRef db="intact" dbAc="MI:0469" id="EBI-619654"
189             <secondaryRef db="pubmed" dbAc="MI:0446" id="14755292" re
190             <secondaryRef db="so" dbAc="MI:0601" id="SO:0000358" refT
191           </xref>
192         </interactorType>
193         <organism ncbiTaxId="559292">
194           <names>
195             <shortLabel>yeast</shortLabel>
196             <fullName>Saccharomyces cerevisiae</fullName>
197             <alias type="synonym" typeAc="MI:1041">Baker's yeast</alia
198           </names>
199         </organism>
200         <sequence>MLSRIVSNNATRSVMCHQAQVGILYKTNPVRTYATLKEVEMRLKSIKNIEKI
201         <attributeList>
202           <attribute name="crc64">ADC71F3C1E0CDF91</attribute>
203           <attribute name="crc64">ADC71F3C1E0CDF91</attribute>
204         </attributeList>
205       </interactor>
206       <interactor id="4">
207         <names>
208           <shortLabel>atp5e_yeast</shortLabel>
209           <fullName>ATP synthase subunit epsilon, mitochondrial</fullN
210           <alias type="gene name" typeAc="MI:0301">ATP15</alias>
211           <alias type="orf name" typeAc="MI:0306">P0345</alias>
212           <alias type="locus name" typeAc="MI:0305">YPL271W</alias>
```

MIF File Parsing

Interaction Evidence Information (I)

```python
evidList = []

#find evidence
mifEvList = mifTree.xpath( "//m:interactionList/m:interaction",
                           namespaces=mifNs)
for ev in mifEvList:

    evid = {}
    evidList.append( evid )
    expt = None

    # find experiment
    evExpRef = ev.xpath( ".//m:experimentRef/text()",
                         namespaces=mifNs )
    if evExpRef:  # compact MIF
        expt = exByRefId[ evExpRef[0] ]
    else:         # expanded MIF
        evExpId = ev.xpath( ".//m:experimentDescription/@id",
                            namespaces=mifNs )
        if evExpId:
            expt = exByRefId[ evExpId[0] ]

    evid['expt'] = expt

    # find interaction type
    evITname = ev.xpath( "./m:interactionType//m:shortLabel/text()",
                         namespaces=mifNs )
    evITid = ev.xpath( "./m:interactionType/m:xref/*[@db='psi-mi']/@id
                       namespaces=mifNs )

    evid['IntType'] = { 'name': evITname[0],
                        'cv': 'psi-mi',
                        'id': evITid[0] }

    ptList = []
    evid['ptList'] = ptList
```

```
1023    <interactionList>
1024      <interaction id="17">
1025        <names>
1026          <shortLabel>atp14-atp18-1</shortLabel>
1027        </names>
1028        <xref>
1029          <primaryRef db="wwpdb" dbAc="MI:0805" id="6CP3" refType="id
1030          <secondaryRef db="wwpdb" dbAc="MI:0805" id="6CP5" refType="
1031          <secondaryRef db="emdb" dbAc="MI:0936" id="EMD-7548" refTyp
1032          <secondaryRef db="emdb" dbAc="MI:0936" id="EMD-7549" refTyp
1033          <secondaryRef db="emdb" dbAc="MI:0936" id="EMD-7546" refTyp
1034          <secondaryRef db="emdb" dbAc="MI:0936" id="EMD-7547" refTyp
1035          <secondaryRef db="wwpdb" dbAc="MI:0805" id="6CP6" refType="
1036          <secondaryRef db="wwpdb" dbAc="MI:0805" id="6CP7" refType="
1037          <secondaryRef db="intact" dbAc="MI:0469" id="EBI-20595952"
1038        </xref>
1039        <experimentList>
1040          <experimentRef>2</experimentRef>
1041        </experimentList>
1042        <participantList>
1043          <participant id="18">
1095          <participant id="19">
1147          <participant id="20">
1199          <participant id="21">
1251          <participant id="22">
1303          <participant id="23">
1355          <participant id="24">
1407          <participant id="25">
1459          <participant id="26">
1567          <participant id="28">
1619          <participant id="29">
1671          <participant id="30">
1723          <participant id="31">
1775          <participant id="32">
1827        </participantList>
1828        <interactionType>
1829          <names>
1830            <shortLabel>physical association</shortLabel>
1831            <fullName>physical association</fullName>
1832          </names>
1833          <xref>
1834            <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0915" refTy
1835            <secondaryRef db="intact" dbAc="MI:0469" id="EBI-1813147"
1836            <secondaryRef db="pubmed" dbAc="MI:0446" id="14755292" re
1837          </xref>
```

MIF File Parsing

Interaction Evidence Information (II)

```python
    ptList = []
    evid['ptList'] = ptList

    # find participants
    evParts = ev.xpath( ".//m:participantList/m:participant",
                        namespaces=mifNs )

    for part in evParts:
        cpart = {}
        ptList.append(cpart)

        # find interactor
        interactor = None
        intRefId = part.xpath( "./m:interactorRef/text()",
                               namespaces=mifNs )
        if intRefId:  # compact MIF
            interactor = irByRefId[ intRefId[0] ]
        else:         # expanded MIF
            intId = part.xpath( ".//m:interactor/@id",
                                namespaces=mifNs )
            if intId:
                interactor = irByRefId[ intId[0] ]
        cpart['interactor'] = interactor

        # experimental/biological role
        # (bait, prey, ancillary, enzyme, substrate, etc.)
        partRoleList = []
        cpart['roleList'] = partRoleList
        roleList = part.xpath( ".//*[local-name(.)='experimentalRole'
                               "     local-name(.)='biologicalRole'  ]
                               namespaces=mifNs )
        for r in roleList:
            roleName = r.xpath( "./m:names/m:shortLabel/text()",
                                namespaces=mifNs )
            roleId = r.xpath( "./m:xref/*[@db='psi-mi']/@id",
                              namespaces=mifNs )

```

```
<participantList>
  <participant id="18">
    <interactorRef>15</interactorRef>
    <biologicalRole>
      <names>
        <shortLabel>unspecified role</shortLabel>
        <fullName>unspecified role</fullName>
      </names>
      <xref>
        <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0499" r
        <secondaryRef db="intact" dbAc="MI:0469" id="EBI-7778
        <secondaryRef db="pubmed" dbAc="MI:0446" id="14755292
      </xref>
    </biologicalRole>
    <experimentalRoleList>
      <experimentalRole>
        <names>
          <shortLabel>neutral component</shortLabel>
          <fullName>neutral component</fullName>
        </names>
        <xref>
          <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0497"
          <secondaryRef db="intact" dbAc="MI:0469" id="EBI-55
          <secondaryRef db="pubmed" dbAc="MI:0446" id="147552
        </xref>
      </experimentalRole>
    </experimentalRoleList>
    <experimentalPreparationList>
      <experimentalPreparation>
        <names>
          <shortLabel>purified</shortLabel>
          <fullName>purified</fullName>
        </names>
        <xref>
          <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0350"
          <secondaryRef db="intact" dbAc="MI:0469" id="EBI-15
          <secondaryRef db="pubmed" dbAc="MI:0446" id="147552
        </xref>
      </experimentalPreparation>
    </experimentalPreparationList>
    <hostOrganismList>
      <hostOrganism ncbiTaxId="559292">
        <names>
          <shortLabel>yeast</shortLabel>
          <fullName>Saccharomyces cerevisiae</fullName>
```

MIF File Parsing

Interaction Evidence Information (III)

mifFileRead.py

```python
    for part in evParts:
        cpart = {}
        ptList.append(cpart)

        # find interactor
        interactor = None
        intRefId = part.xpath( "./m:interactorRef/text()",
                               namespaces=mifNs )
        if intRefId:  # compact MIF
            interactor = irByRefId[ intRefId[0] ]
        else:         # expanded MIF
            intId = part.xpath( ".//m:interactor/@id",
                                namespaces=mifNs )
            if intId:
                interactor = irByRefId[ intId[0] ]
        cpart['interactor'] = interactor

        # experimental/biological role
        # (bait, prey, ancillary, enzyme, substrate, etc.)
        partRoleList = []
        cpart['roleList'] = partRoleList
        roleList = part.xpath( ".//*[local-name(.)='experimentalRole'
                               "     local-name(.)='biologicalRole'  ]
                               namespaces=mifNs )
        for r in roleList:
            roleName = r.xpath( "./m:names/m:shortLabel/text()",
                                namespaces=mifNs )
            roleId = r.xpath( "./m:xref/*[@db='psi-mi']/@id",
                              namespaces=mifNs )

            if roleName[0] != 'unspecified role':
                role = { 'name': roleName[0],
                         'cv': 'psi-mi',
                         'id': roleId[0] }
                partRoleList.append(role)
import json
print( json.dumps( evidList, sort_keys=True, indent=4) )
```

```
<participantList>
  <participant id="18">
    <interactorRef>15</interactorRef>
    <biologicalRole>
      <names>
        <shortLabel>unspecified role</shortLabel>
        <fullName>unspecified role</fullName>
      </names>
      <xref>
        <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0499"
        <secondaryRef db="intact" dbAc="MI:0469" id="EBI-778
        <secondaryRef db="pubmed" dbAc="MI:0446" id="1475292
      </xref>
    </biologicalRole>
    <experimentalRoleList>
      <experimentalRole>
        <names>
          <shortLabel>neutral component</shortLabel>
          <fullName>neutral component</fullName>
        </names>
        <xref>
          <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0497"
          <secondaryRef db="intact" dbAc="MI:0469" id="EBI-55
          <secondaryRef db="pubmed" dbAc="MI:0446" id="147552
        </xref>
      </experimentalRole>
    </experimentalRoleList>
    <experimentalPreparationList>
      <experimentalPreparation>
        <names>
          <shortLabel>purified</shortLabel>
          <fullName>purified</fullName>
        </names>
        <xref>
          <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0350"
          <secondaryRef db="intact" dbAc="MI:0469" id="EBI-15
          <secondaryRef db="pubmed" dbAc="MI:0446" id="147552
        </xref>
      </experimentalPreparation>
    </experimentalPreparationList>
    <hostOrganismList>
      <hostOrganism ncbiTaxId="559292">
        <names>
          <shortLabel>yeast</shortLabel>
          <fullName>Saccharomyces cerevisiae</fullName>
```

MIF File Parsing

Merging Interaction Data (I)

mifFileMerge.py

```python
#!/usr/bin/python3

import sys
import json

from os import walk
from os.path import join

from lxml import etree

mifNs = { 'm': 'http://psi.hupo.org/mi/mif'}

def getSource( mifData, mifTree ):

    if 'sourceList' not in mifData:
        mifData['sourceList'] = []

    # find record source
    mifSrcName = mifTree.xpath( "//m:source//m:shortLabel/text()",
                                namespaces=mifNs)
    # find record release date
    mifDate = mifTree.xpath( "//m:source/@releaseDate",
                             namespaces=mifNs)

    mifData['sourceList'].append( { "database": mifSrcName[0],
                                    "date": mifDate[0] } )
    return mifData


def getExperiments( mifData, mifTree ):

    mifData['exByRefId'] = {}

    #find experiments
    mifExList = mifTree.xpath( "//m:experimentDescription",
                               namespaces=mifNs)
    for ex in mifExList:
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<entrySet xmlns="http://psi.hupo.org/mi/mif" xmlns:xsi="http://www.w3.o
  <entry>
    <source releaseDate="2019-05-13">
      <names>
        <shortLabel>DIP</shortLabel>
        <fullName>Database of Interacting Proteins</fullName>
      </names>
      <xref>
        <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0465" refType="id
        <secondaryRef db="pubmed" dbAc="MI:0446" id="14681454" refType=
        <secondaryRef db="intact" dbAc="MI:0469" id="EBI-1579232" refTy
      </xref>
      <attributeList>
        <attribute name="postaladdress">611 Young Drive East; Los Angel
        <attribute name="url" nameAc="MI:0614">http://dip.doe-mbi.ucla.
        <attribute name="contact-email" nameAc="MI:0634">dip@mbi.ucla.e
      </attributeList>
    </source>
    <experimentList>
      <experimentDescription id="1">
      <experimentDescription id="2">
    </experimentList>
    <interactorList>
      <interactor id="3">
      <interactor id="4">
      <interactor id="5">
      <interactor id="6">
      <interactor id="7">
      <interactor id="8">
      <interactor id="9">
      <interactor id="10">
      <interactor id="11">
      <interactor id="12">
      <interactor id="13">
      <interactor id="14">
      <interactor id="15">
      <interactor id="16">
    </interactorList>
    <interactionList>
      <interaction id="17">
      <interaction id="33">
    </interactionList>
  </entry>
</entrySet>
```

MIF File Parsing

Merging Interaction Data (II)

mifFileMerge.py

```python
mifData = {}

assert len(sys.argv) == 4
mydir = sys.argv[1]
mymth = sys.argv[2]
myout = sys.argv[3]

myfiles = []
for (dirpath, dirnames, filenames) in walk( mydir ):
    myfiles.extend(filenames)
    break

for f in myfiles:
    mifTree = etree.parse( join( mydir, f ) )

    getSource( mifData, mifTree )
    getExperiments( mifData, mifTree )
    getInteractors( mifData, mifTree )
    getEvidence( mifData, mifTree )
    dropInternalRefs( mifData )

#-------------------------------------------------
# operations
#------------

if myout == 'json':
    print( toJson(mifData, indent=1) )

if mymth in ['spoke','matrix']:
    network = buildNetwork( mifData, expand=mymth

if myout == 'sif':
    print( toSif( mifData, network) )
elif myout == 'nodeprop':
    print( toNodeProperties( mifData, network) )
elif myout == 'edgeprop':
    print( toEdgeProperties( mifData, network) )
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<entrySet xmlns="http://psi.hupo.org/mi/mif" xmlns:xsi="http://www.w3.o
  <entry>
    <source releaseDate="2019-05-13">
      <names>
        <shortLabel>DIP</shortLabel>
        <fullName>Database of Interacting Proteins</fullName>
      </names>
      <xref>
        <primaryRef db="psi-mi" dbAc="MI:0488" id="MI:0465" refType="id
        <secondaryRef db="pubmed" dbAc="MI:0446" id="14681454" refType=
        <secondaryRef db="intact" dbAc="MI:0469" id="EBI-1579232" refTy
      </xref>
      <attributeList>
        <attribute name="postaladdress">611 Young Drive East; Los Angel
        <attribute name="url" nameAc="MI:0614">http://dip.doe-mbi.ucla.
        <attribute name="contact-email" nameAc="MI:0634">dip@mbi.ucla.e
      </attributeList>
    </source>
    <experimentList>
      <experimentDescription id="1">
      <experimentDescription id="2">
    </experimentList>
    <interactorList>
```

stackoverflow | Products | Customers | Use cases | How do I list all files of a directory | Log in | Sign up

Home
PUBLIC
Stack Overflow
Tags
Users
TEAMS
Free 30 Day Trial

3868

`os.listdir()` will get you everything that's in a directory - files and directories.

If you want *just* files, you could either filter this down using `os.path`:

```
from os import listdir
from os.path import isfile, join
onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
```

or you could use `os.walk()` which will yield two lists for each directory it visits - splitting into files and dirs for you. If you only want the top directory you can just break the first time it yields

```
from os import walk

f = []
for (dirpath, dirnames, filenames) in walk(mypath):
    f.extend(filenames)
    break
```

share | improve this answer | edited Jun 5 '19 at 3:10 cs95 189k • 38 • 294 • 358 | answered Jul 8 '10 at 21:01 pycruft 41.8k • 1 • 14 • 10

Linked

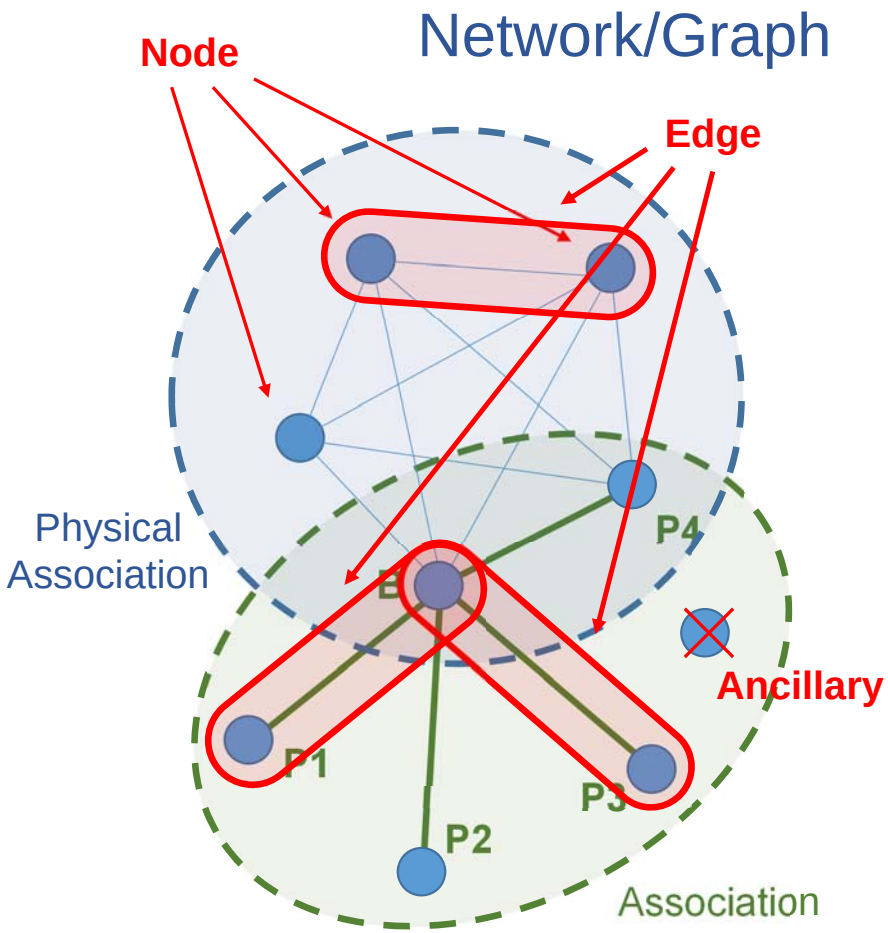
- 16 How to get only files in directory?
- 10 Reading all files in all directories
- 11 Python3 list files from particular directory
- 14 python : get list all *.txt files in a directory
- -5 print the directory for files in Python
- 4 How to list directory using Python
- 1 How can I automatically open all text files in a given folder?
- 2 How do i use Linux terminal commands like CD and LS?
- 2 How to get file path + file name into a list?
- 2 Automatically creating a list in Python

see more linked questions...

MIF File Parsing

Merging Interaction Data

```python
def buildNetwork( mifData, expand=None):

    # expand:
    # None    - binary edges only
    # spoke   - bait/prey (spoke) expansion
    # matrix - spoke + matrix of physical and below, ignoring ancillar

    # network data (note: participant details are losts !!!)
    #   key:    (acc1,acc2);
    #   value: {'evidList': [ev1,ev2,...], 'intType':[...]}
    edges = {}

    for ev in mifData['evidList']:
        bait = None
        preyDict = {}
        partDict = {}
        for p in ev['partList']:
            skip = False
            for r in p['roleList']:
                if r['name'] == 'bait':
                    bait = p['interactor']['acc']
                if r['name'] == 'prey':
                    if p['interactor']['acc'] in preyDict:
                        preyDict[ p['interactor']['acc'] ] += 1
                    else:
                        preyDict[ p['interactor']['acc'] ] = 1

                if r['name'] == 'ancillary':
                    skip = True
            if not skip:
                if p['interactor']['acc'] in partDict:
                    partDict[ p['interactor']['acc'] ] += 1
                else:
                    partDict[ p['interactor']['acc'] ] = 1

        if len(partDict) == 2:
```



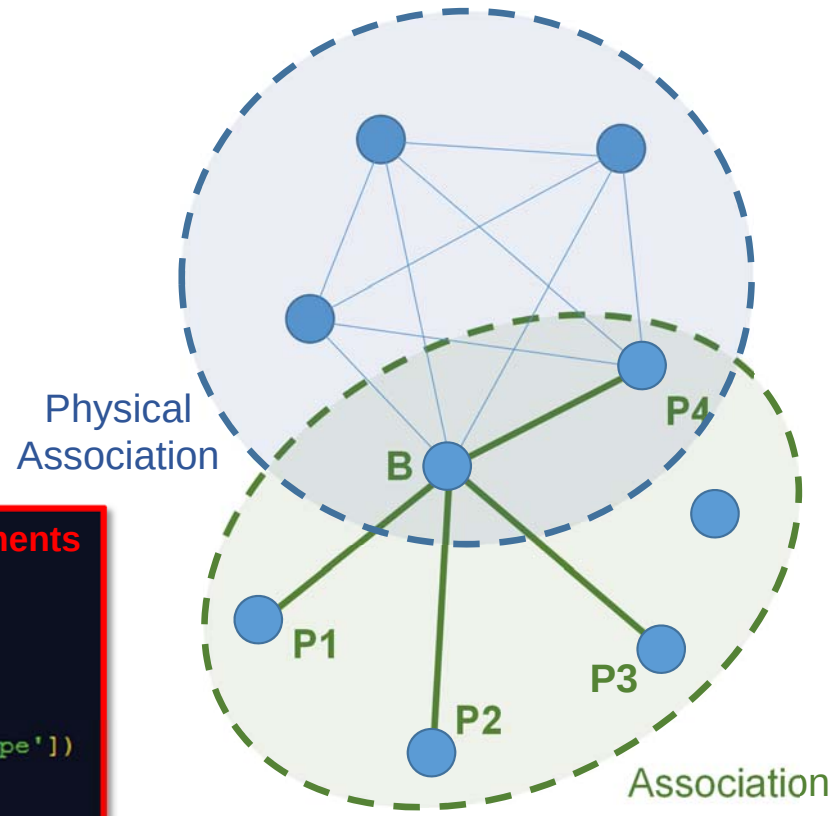
MIF File Parsing

Merging Interaction Data

```python
def buildNetwork( mifData, expand=None):

    # expand:
    # None    - binary edges only
    # spoke   - bait/prey (spoke) expansion
    # matrix - spoke + matrix of physical and below, ignoring ancillar

    # network data (note: participant details are losts !!!)
    #   key:    (acc1,acc2);
    #   value: {'evidList': [ev1,ev2,...], 'intType':[...]}
    edges = {}

    for ev in mifData['evidList']:
        bait = None
        preyDict = {}
        partDict = {}
        for p in ev['partList']:
            skip = False
            for r in p['roleList']:

                if len(partDict) == 2:
                    (part1, part2) = ( partDict.keys() )
                    if part1 > part2:
                        (part2, part1) = (part1, part2)
                    if (part1, part2) in edges:
                        edges[(part1, part2)]['evidList'].append(ev )
                        edges[(part1, part2)]['intType'].append(ev['intType'])

                    elif part1 != part2:
                        edges[(part1, part2)] = {'evidList': [ ev ],
                                                 'intType': [ev['intType']]}


        if len(partDict) == 2:
```

'Binary' Experiments

Network/Graph

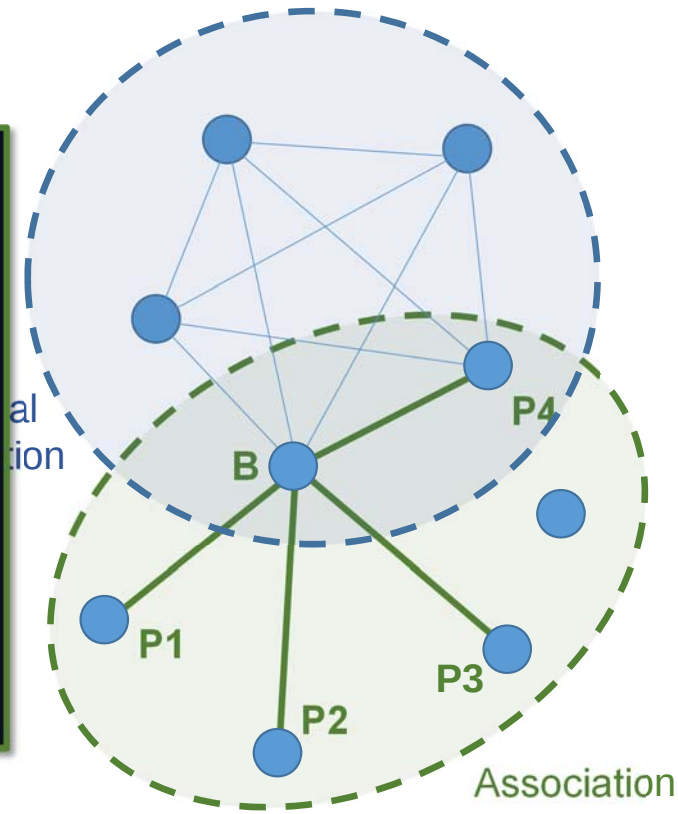


MIF File Parsing

Merging Interaction Data

```
def buildNetwork( mifData, expand=None):

    # expand:
    # None    - binary edges only
    # spoke   - bait/prey (spoke) expansion

    elif len(partDict) > 2:
        if expand in ['spoke','matrix']:
            if bait is not None:
                for prey in preyDict.keys():

                    itype = {'cv': 'psi-mi', 'id': 'MI:0915',
                             'name': 'physical association' }

                    if bait > prey:
                        (part1, part2) = (prey, bait)
                    else:
                        (part1, part2) = (bait, prey)
                    if (part1, part2) in edges:
                        edges[(part1, part2)]['evidList'].append( ev )
                        edges[(part1, part2)]['intType'].append( itype )
                    elif part1 != part2:
                        edges[ (part1, part2) ] = { 'evidList': [ ev ],
                                                    'intType': [itype] }

                    skip = True
            if not skip:
                if p['interactor']['acc'] in partDict:
                    partDict[ p['interactor']['acc'] ] += 1
                else:
                    partDict[ p['interactor']['acc'] ] = 1

        if len(partDict) == 2:
```

Network/Graph



MIF File Parsing

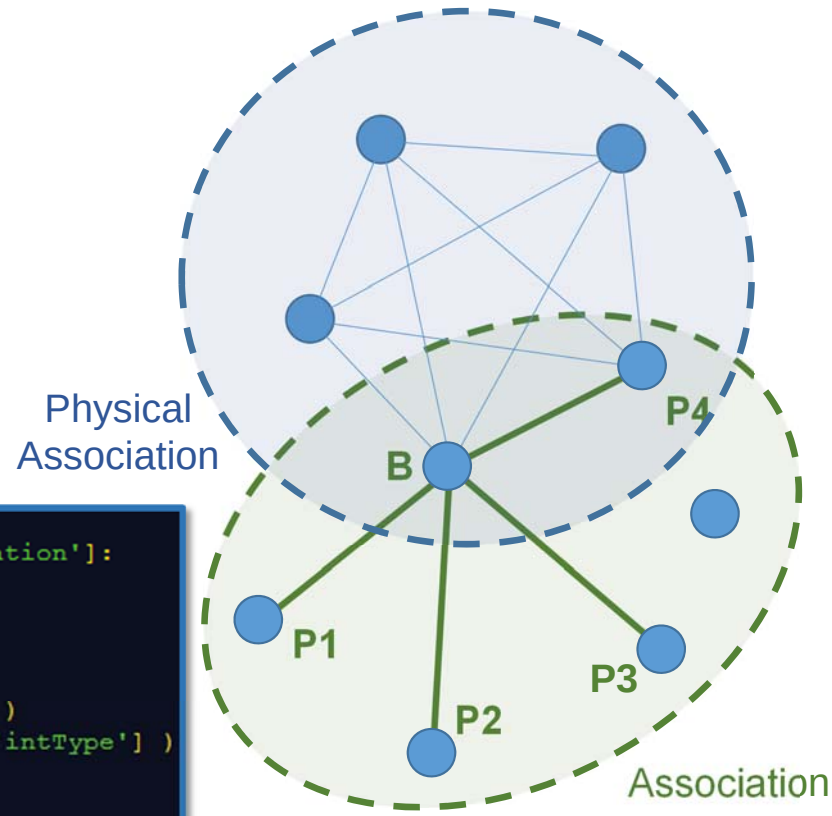
Merging Interaction Data

```python
def buildNetwork( mifData, expand=None):

    # expand:
    # None    - binary edges only
    # spoke   - bait/prey (spoke) expansion
    # matrix - spoke + matrix of physical and below, ignoring ancillar

    # network data (note: participant details are losts !!!)
    #   key:    (acc1,acc2);
    #   value: {'evidList': [ev1,ev2,...], 'intType':[...]}
    edges = {}

    for ev in mifData['evidList']:
        bait = None
        preyDict = {}
        partDict = {}
        for p in ev['partList']:
            skip = False
            for r in p['roleList']:
```

```python
if expand in ['matrix'] and ev['intType']['name'] not in ['association']:
    for part1 in partDict.keys():
        for part2 in partDict.keys():
            if part2 > part1:
                if (part1, part2) in edges:
                    edges[ (part1,part2) ]['evidList'].append( ev )
                    edges[ (part1,part2) ]['intType'].append( ev['intType'] )
                else:
                    edges[ (part1,part2) ] = {'evidList': [ ev ],
                                              'intType': [ev['intType']] }
```

```python
                partDict[ p['interactor']['acc'] ] = 1

        if len(partDict) == 2:
```

Network/Graph



MIF File Parsing

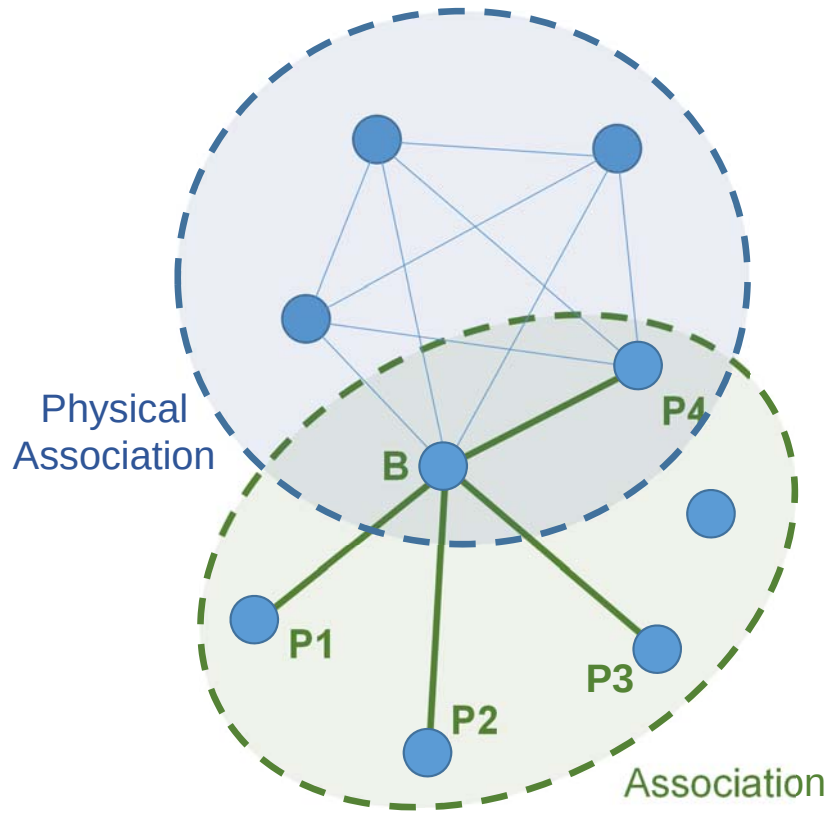
Merging Interaction Data

```python
def toJson( mifData,  indent=None ):
    return json.dumps( mifData, sort_keys=True, indent=indent )


def toSif( mifData, network ):
    buffer = ''
    for edge in network:
        (p1,p2) = edge

        p1type = mifData['irByAccId'][p1]['type']['name'][0]
        p2type = mifData['irByAccId'][p2]['type']['name'][0]

        buffer += "%s %s%s %s\n" % ( p1, p1type, p2type, p2 )
    return buffer


def toEdgePoperties( mifData, network ):
    buffer = '#acc\ttype id\ttype name\tmethod id\tmethod name\tpmid\n'
    for edge in network:
        (p1,p2) = edge
        evList = network[edge]['evidList']
        itypeList = network[edge]['intType']

        p1type = mifData['irByAccId'][p1]['type']['name'][0]
        p2type = mifData['irByAccId'][p2]['type']['name'][0]

        line = "%s (%s%s) %s\t" % (p1, p1type, p2type, p2)

        #  type
        typeId=''
        typeName=''
        for itype in itypeList:
            typeId += itype['id'] + '|'
            typeName += itype['name'] + '|'

        line += typeId[:-1] + "\t" +  typeName[:-1] +"\t"

        # method
        methodId=''
        methodName=''
        for ev in evList:
            methodId +=   ev['expt']['detMeth']['id'] + '|'
            methodName += ev['expt']['detMeth']['name'] + '|'

        line += methodId[:-1] + "\t" +  methodName[:-1] + "\t"

        # source
        source = ''
```

JSON

SIF

Network/Graph



MIF File Parsing

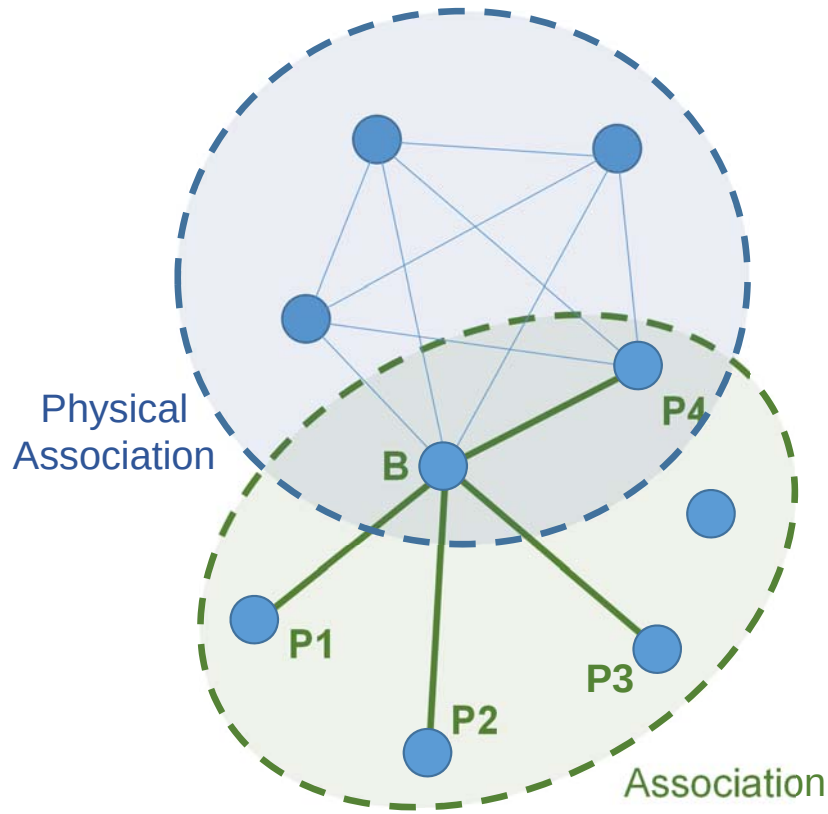
Merging Interaction Data

```
def toEdgePoperties( mifData, network ):
    buffer = '#acc\ttype id\ttype name\tmethod id\tmethod name\tpmid\n'
    for edge in network:
        (p1,p2) = edge
        evList = network[edge]['evidList']
        itypeList = network[edge]['intType']

        p1type = mifData['irByAccId'][p1]['type']['name'][0]
        p2type = mifData['irByAccId'][p2]['type']['name'][0]

        line = "%s (%s%s) %s\t" % (p1, p1type, p2type, p2)

        #  type
        typeId=''
        typeName=''
        for itype in itypeList:
            typeId += itype['id'] + '|'
            typeName += itype['name'] + '|'

        line += typeId[:-1] + "\t" +  typeName[:-1] +"\t"

        # method
        methodId=''
        methodName=''
        for ev in evList:
            methodId +=   ev['expt']['detMeth']['id'] + '|'
            methodName += ev['expt']['detMeth']['name'] + '|'

        line += methodId[:-1] + "\t" +  methodName[:-1] + "\t"

        # source
        source = ''
        for ev in evList:
            source += ev['expt']['pmid'] + '|'

        line += source[:-1] + "\t"

        buffer += line + "\n"

    return buffer


def toNodePoperties( mifData, network ):
    buffer = '#acc\tlabel\ttype\ttaxid\ttaxon label\ttaxon name\tgene\n'

    nodeDict = mifData['irByAccId']

    for node in nodeDict:
```

Edge Identifier

Edge Properties

Network/Graph



MIF File Parsing

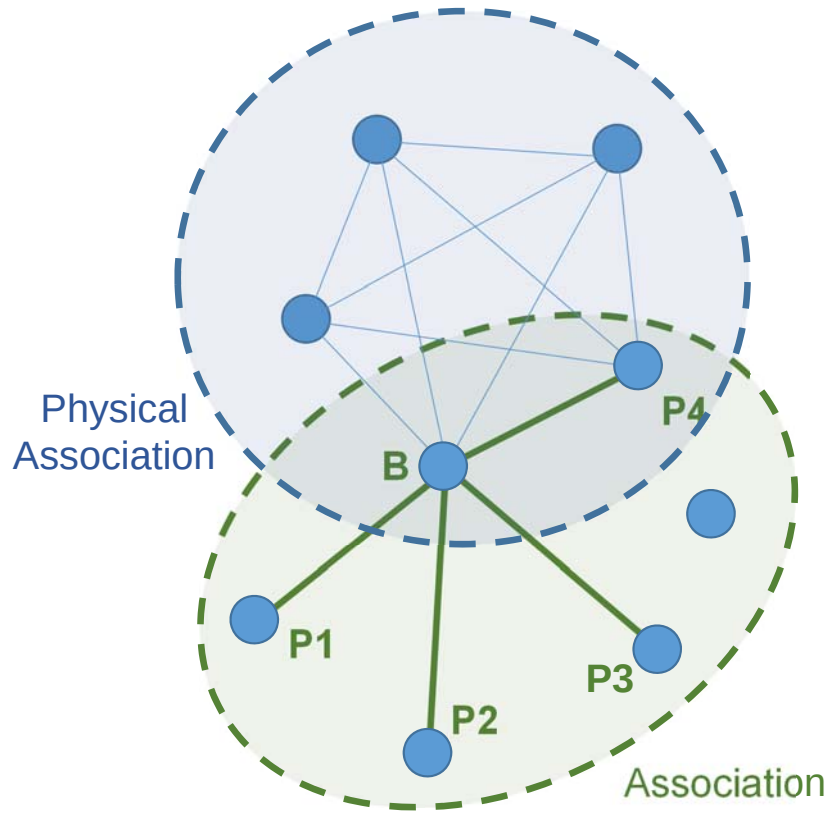
Merging Interaction Data

```
def toNodePoperties( mifData, network ):
    buffer = '#acc\tlabel\ttype\ttaxid\ttaxon label\ttaxon name\tgene\n'

    nodeDict = mifData['irByAccId']

    for node in nodeDict:
        #acession
        line = "%s\t" % ( nodeDict[node]['acc'] )

        # label
        line += "%s\t" %( nodeDict[node]['label'] )

        # type
        line += "%s\t" %( nodeDict[node]['type']['name'] )

        # taxid
        line += "%s\t" %( nodeDict[node]['taxon']['taxid'] )

        # taxon label
        line += "%s\t" %( nodeDict[node]['taxon']['label'] )

        # taxon name
        line += "%s\t" %( nodeDict[node]['taxon']['name'] )

        # gene
        line += "%s\t" %( nodeDict[node]['gene'] )

        buffer += line + "\n"
    return buffer


#-----------------------------------------------------------------------
# merge files
#------------

mifData = {}

assert len(sys.argv) == 4
mydir = sys.argv[1]
mymth = sys.argv[2]
myout = sys.argv[3]

myfiles = []
for (dirpath, dirnames, filenames) in walk( mydir ):
    myfiles.extend(filenames)
    break

for f in myfiles:
```

Node Identifier

Node Properties

Network/Graph



Interaction Data Processing

Take Home Message(s)

- XML parsing is not that hard !!!
  - In many cases it is much easier that parsing text files
- Use example scripts as a starting point for your own
  - To read UniprotKB records
  - To read PSI-MI XML interaction files
  - To read any other XML (or HTML) files
- Be careful when dealing with
  - Potentially ambiguous and/or inconsistent protein identifiers
  - Interaction types
  - Multi-protein interactions (spoke/matrix expansion, changing interaction types)

Interaction Data Processing

XML file parsing

Simple (and less simple but somewhat useful) project ideas

- List UniprotKB (gene names, Entrez gene identifiers, GO terms, etc) identifiers for each protein in a given MIF file
- Generate a FASTA file listing sequences all the bait proteins reported in a given MIF file
- Count interactions of a given protein (or a set of proteins) that are reported in a given MIF file
- Find all direct interactions of a given protein that are reported in a given MIF file
- List all proteins for each interaction reported in a given MIF file excluding proteins annotated (experimental role) as 'ancillary'
- Retrieve GO annotation from XML-formatted UniprotKB record – e.g.

  https://www.uniprot.org/uniprot/P60010.xml

Interaction Data Processing

References

XML Parsing

- lxml library: https://lxml.de
- XPath: https://en.wikipedia.org/wiki/XPath (and references therein)

MIF (miXML) format specification

- XSD: https://github.com/HUPO-PSI/miXML (2.5/src, 3.0/src directories)
- Publications
  - Hermjakob H *et al*. The HUPO PSI's molecular interaction format--a community standard for the representation of protein interaction data.

    Nat Biotechnol. 22:177-83 (2004). PMID: 14755292
  - Kerrien S et al. Broadening the horizon--level 2.5 of the HUPO-PSI format for molecular interactions.

    BMC Biol. 5:44 (2007). PMID: 17925023
  - Sivade Dumousseau M *et al*. Encompassing new use cases - level 3.0 of the HUPO-PSI format for molecular interactions.

    BMC Bioinformatics. 19:134. doi: 10.1186/s12859-018-2118-1 (2018). PMID: 29642841