`gPy` User Manual

James Cussens
University of York

21/10/2007

Abstract

gPy is a Python package intended to help students (and others) understand algorithms related to graphical models, particularly Bayesian networks. Python was chosen since it is easy to learn, high-level and because it is 'executable pseudo-code'. Why present an algorithm using non-executable pseudo-code when an executable version is possible? The object-oriented features of Python are used to reinforce commonalities between different graphical models.

gPy does not contain any algorithms not found elsewhere and since it is implemented in an interpreted language it will not provide the fastest implementation of those algorithms.

The only thing you need to have to use gPy is an installation of Python 2.4 or higher. If you're running a reasonably recent distribution of Linux, Python 2.4 will probably already be available. In any event, the latest version of Python (for Linux, Windows and Mac) can be downloaded from `www.python.com`

Contents

Chapter 1

How to read this manual

This manual contains many examples of using `gPy` in a Python interactive session. To do these examples yourself, just start the Python interpreter and type in the stuff (displayed `in this font`) which comes after the prompt `>>>` . Stuff in normal font is my commentary on what you are doing, so does not provide it as input to the Python interpreter.

Many examples produce output (also `in this font`), and the Python module `doctest` has been used to check that the output you see here is what is actually produced. There is one small exception. Sometimes the output does not fit on one line of text, so line breaks have been added to maintain readability. In such cases the text `<FAKE_NL>` has been added to help the checking script; you, the reader, should just ignore it. Note that no attempt has been made to fit each Python examples on a single page.

Chapter 2

Quick start

Assuming that you have installed `gPy` (and Python!) so that your Python installation can find the `gPy` modules, you can do the following to get a quick idea of how to use `gPy`. Firstly, grab the *Asia* Bayesian network from the `gPy.Examples` modules. This network has the name `asia` in that module so you do:

```
>>> from gPy.Examples import asia
```

Check that the name `asia` really does refer to an object in the class `BN`.

```
>>> type(asia)
<class 'gPy.Models.BN'>
```

The obvious thing to do now is to have a look at the BN.

```
>>> print asia

Smoking   |     Bronchitis
          |   absent  present
--------- | -----------------
nonsmoker |     0.70     0.30
smoker    |     0.40     0.60


Smoking   |       Cancer
          |   absent  present
--------- | -----------------
nonsmoker |     0.99     0.01
smoker    |     0.90     0.10


Bronchitis | TbOrCa |      Dyspnea
           |        |   absent  present
```

```
---------- | ------ | ----------------
absent     | false  |     0.90     0.10
absent     | true   |     0.30     0.70
present    | false  |     0.20     0.80
present    | true   |     0.10     0.90


      Smoking
  nonsmoker  smoker
-------------------
       0.50    0.50


Cancer  | Tuberculosis |     TbOrCa
        |              |   false  true
------- | ------------ | -------------
absent  | absent       |    1.00  0.00
absent  | present      |    0.00  1.00
present | absent       |    0.00  1.00
present | present      |    0.00  1.00


VisitAsia |    Tuberculosis
          |   absent  present
--------- | -----------------
no_visit  |     0.99     0.01
visit     |     0.95     0.05


    VisitAsia
  no_visit  visit
-----------------
      0.99   0.01


TbOrCa |        XRay
       |   abnormal  normal
------ | ------------------
false  |       0.05    0.95
true   |       0.98    0.02
```

So, the textual representation of a BN is simply a list of the CPTs in the BN. Each CPT in the BN can be accessed via the name of its child variable.

```
>>> print asia['Bronchitis']
```

```
Smoking   |     Bronchitis
          |   absent  present
--------- | -----------------
nonsmoker |     0.70     0.30
smoker    |     0.40     0.60

>>> print asia['Smoking']

      Smoking
  nonsmoker  smoker
-------------------
       0.50    0.50
```

A basic operation for graphical models is *factor multiplication*, CPTs are a special sort of factor. To multiply two CPTs we just use the `*` operator.

```
>>> example_factor = asia['Bronchitis'] * asia['Smoking']
>>> print example_factor

Bronchitis | Smoking   |
---------- | --------- | ----
absent     | nonsmoker |   0.35
absent     | smoker    |   0.20
present    | nonsmoker |   0.15
present    | smoker    |   0.30
```

`example_factor` is a `Factor` object, whereas e.g. `asia['Bronchitis']` is a `CPT` object (a special case—a subclass—of a `Factor` object). This is why they are printed out differently.

```
>>> type(example_factor)
<class 'gPy.Parameters.Factor'>
>>> type(asia['Bronchitis'])
<class 'gPy.Parameters.CPT'>
```

Factors are functions mapping combinations of values of discrete variables to numbers. These numbers don't have to probabilities.

```
>>> example_factor *= 5
>>> print example_factor

Bronchitis | Smoking   |
---------- | --------- | ----
absent     | nonsmoker |   1.75
absent     | smoker    |   1.00
```

```
present    | nonsmoker |   0.75
present    | smoker    |   1.50
```

Note that we just altered the object `example_factor` by multiplying it by 5. Many `gPy` operations alter objects so it is sometimes useful to make copies.

```
>>> my_copy = example_factor.copy()
```

Simple numbers are allowed to be treated as factors. Also we can iterate over all the CPTs in a BN. These two facts make it easy to make a factor which is an explicit representation of the full joint distribution defined by `asia`. If you're not familiar with using the Python interpreter be careful: On the first line with ...put a space so that the `j` of `joint` lines up with the `o` of `for`. This provides white space indentation which Python uses to make program blocks. Just hit return when you get the second ...prompt.

```
>>> joint = 1
>>> for cpt in asia:
...  joint *= cpt
...
>>> print joint

Bronchitis | Cancer  | Dyspnea | Smoking   | TbOrCa | Tuberculosis | VisitAsia | XRay     |
---------- | ------- | ------- | --------- | ------ | ------------ | --------- | -------- | ----
absent     | absent  | absent  | nonsmoker | false  | absent       | no_visit  | abnormal |   0.02
absent     | absent  | absent  | nonsmoker | false  | absent       | no_visit  | normal   |   0.29
absent     | absent  | absent  | nonsmoker | false  | absent       | visit     | abnormal |   0.00
absent     | absent  | absent  | nonsmoker | false  | absent       | visit     | normal   |   0.00
absent     | absent  | absent  | nonsmoker | false  | present      | no_visit  | abnormal |   0.00
absent     | absent  | absent  | nonsmoker | false  | present      | no_visit  | normal   |   0.00
absent     | absent  | absent  | nonsmoker | false  | present      | visit     | abnormal |   0.00
absent     | absent  | absent  | nonsmoker | false  | present      | visit     | normal   |   0.00
absent     | absent  | absent  | nonsmoker | true   | absent       | no_visit  | abnormal |   0.00
absent     | absent  | absent  | nonsmoker | true   | absent       | no_visit  | normal   |   0.00
absent     | absent  | absent  | nonsmoker | true   | absent       | visit     | abnormal |   0.00
absent     | absent  | absent  | nonsmoker | true   | absent       | visit     | normal   |   0.00
absent     | absent  | absent  | nonsmoker | true   | present      | no_visit  | abnormal |   0.00
absent     | absent  | absent  | nonsmoker | true   | present      | no_visit  | normal   |   0.00
absent     | absent  | absent  | nonsmoker | true   | present      | visit     | abnormal |   0.00
absent     | absent  | absent  | nonsmoker | true   | present      | visit     | normal   |   0.00
absent     | absent  | absent  | smoker    | false  | absent       | no_visit  | abnormal |   0.01
absent     | absent  | absent  | smoker    | false  | absent       | no_visit  | normal   |   0.15
absent     | absent  | absent  | smoker    | false  | absent       | visit     | abnormal |   0.00
absent     | absent  | absent  | smoker    | false  | absent       | visit     | normal   |   0.00
absent     | absent  | absent  | smoker    | false  | present      | no_visit  | abnormal |   0.00
absent     | absent  | absent  | smoker    | false  | present      | no_visit  | normal   |   0.00
absent     | absent  | absent  | smoker    | false  | present      | visit     | abnormal |   0.00
absent     | absent  | absent  | smoker    | false  | present      | visit     | normal   |   0.00
absent     | absent  | absent  | smoker    | true   | absent       | no_visit  | abnormal |   0.00
absent     | absent  | absent  | smoker    | true   | absent       | no_visit  | normal   |   0.00
absent     | absent  | absent  | smoker    | true   | absent       | visit     | abnormal |   0.00
absent     | absent  | absent  | smoker    | true   | absent       | visit     | normal   |   0.00
absent     | absent  | absent  | smoker    | true   | present      | no_visit  | abnormal |   0.00
absent     | absent  | absent  | smoker    | true   | present      | no_visit  | normal   |   0.00
absent     | absent  | absent  | smoker    | true   | present      | visit     | abnormal |   0.00
absent     | absent  | absent  | smoker    | true   | present      | visit     | normal   |   0.00
absent     | absent  | present | nonsmoker | false  | absent       | no_visit  | abnormal |   0.00
absent     | absent  | present | nonsmoker | false  | absent       | no_visit  | normal   |   0.03
absent     | absent  | present | nonsmoker | false  | absent       | visit     | abnormal |   0.00
absent     | absent  | present | nonsmoker | false  | absent       | visit     | normal   |   0.00
absent     | absent  | present | nonsmoker | false  | present      | no_visit  | abnormal |   0.00
absent     | absent  | present | nonsmoker | false  | present      | no_visit  | normal   |   0.00
absent     | absent  | present | nonsmoker | false  | present      | visit     | abnormal |   0.00
absent     | absent  | present | nonsmoker | false  | present      | visit     | normal   |   0.00
absent     | absent  | present | nonsmoker | true   | absent       | no_visit  | abnormal |   0.00
absent     | absent  | present | nonsmoker | true   | absent       | no_visit  | normal   |   0.00
absent     | absent  | present | nonsmoker | true   | absent       | visit     | abnormal |   0.00
absent     | absent  | present | nonsmoker | true   | absent       | visit     | normal   |   0.00
absent     | absent  | present | nonsmoker | true   | present      | no_visit  | abnormal |   0.00
absent     | absent  | present | nonsmoker | true   | present      | no_visit  | normal   |   0.00
absent     | absent  | present | nonsmoker | true   | present      | visit     | abnormal |   0.00
absent     | absent  | present | nonsmoker | true   | present      | visit     | normal   |   0.00
absent     | absent  | present | smoker    | false  | absent       | no_visit  | abnormal |   0.00
absent     | absent  | present | smoker    | false  | absent       | no_visit  | normal   |   0.02
absent     | absent  | present | smoker    | false  | absent       | visit     | abnormal |   0.00
absent     | absent  | present | smoker    | false  | absent       | visit     | normal   |   0.00
absent     | absent  | present | smoker    | false | present    | no_visit | abnormal |   0.00
absent     | absent  | present | smoker    | false | present    | no_visit | normal   |   0.00
absent     | absent  | present | smoker    | false | present    | visit    | abnormal |   0.00
absent     | absent  | present | smoker    | false | present    | visit    | normal   |   0.00
absent     | absent  | present | smoker    | true  | absent     | no_visit | abnormal |   0.00
absent     | absent  | present | smoker    | true  | absent     | no_visit | normal   |   0.00
absent     | absent  | present | smoker    | true  | absent     | visit    | abnormal |   0.00
absent     | absent  | present | smoker    | true  | absent     | visit    | normal   |   0.00
absent     | absent  | present | smoker    | true  | present    | no_visit | abnormal |   0.00
absent     | absent  | present | smoker    | true  | present    | no_visit | normal   |   0.00
absent     | absent  | present | smoker    | true  | present    | visit    | abnormal |   0.00
absent     | absent  | present | smoker    | true  | present    | visit    | normal   |   0.00
absent     | present | absent  | nonsmoker | false | absent     | no_visit | abnormal |   0.00
absent     | present | absent  | nonsmoker | false | absent     | no_visit | normal   |   0.00
absent     | present | absent  | nonsmoker | false | absent     | visit    | abnormal |   0.00
absent     | present | absent  | nonsmoker | false | absent     | visit    | normal   |   0.00
absent     | present | absent  | nonsmoker | false | present    | no_visit | abnormal |   0.00
absent     | present | absent  | nonsmoker | false | present    | no_visit | normal   |   0.00
absent     | present | absent  | nonsmoker | false | present    | visit    | abnormal |   0.00
absent     | present | absent  | nonsmoker | false | present    | visit    | normal   |   0.00
absent     | present | absent  | nonsmoker | true  | absent     | no_visit | abnormal |   0.00
absent     | present | absent  | nonsmoker | true  | absent     | no_visit | normal   |   0.00
absent     | present | absent  | nonsmoker | true  | absent     | visit    | abnormal |   0.00
absent     | present | absent  | nonsmoker | true  | absent     | visit    | normal   |   0.00
absent     | present | absent  | nonsmoker | true  | present    | no_visit | abnormal |   0.00
absent     | present | absent  | nonsmoker | true  | present    | no_visit | normal   |   0.00
absent     | present | absent  | nonsmoker | true  | present    | visit    | abnormal |   0.00
absent     | present | absent  | nonsmoker | true  | present    | visit    | normal   |   0.00
absent     | present | absent  | smoker    | false | absent     | no_visit | abnormal |   0.00
absent     | present | absent  | smoker    | false | absent     | no_visit | normal   |   0.00
absent     | present | absent  | smoker    | false | absent     | visit    | abnormal |   0.00
absent     | present | absent  | smoker    | false | absent     | visit    | normal   |   0.00
absent     | present | absent  | smoker    | false | present    | no_visit | abnormal |   0.00
absent     | present | absent  | smoker    | false | present    | no_visit | normal   |   0.00
absent     | present | absent  | smoker    | false | present    | visit    | abnormal |   0.00
absent     | present | absent  | smoker    | false | present    | visit    | normal   |   0.00
absent     | present | absent  | smoker    | true  | absent     | no_visit | abnormal |   0.01
absent     | present | absent  | smoker    | true  | absent     | no_visit | normal   |   0.00
absent     | present | absent  | smoker    | true  | absent     | visit    | abnormal |   0.00
absent     | present | absent  | smoker    | true  | absent     | visit    | normal   |   0.00
absent     | present | absent  | smoker    | true  | present    | no_visit | abnormal |   0.00
absent     | present | absent  | smoker    | true  | present    | no_visit | normal   |   0.00
absent     | present | absent  | smoker    | true  | present    | visit    | abnormal |   0.00
absent     | present | absent  | smoker    | true  | present    | visit    | normal   |   0.00
absent     | present | present | nonsmoker | false | absent     | no_visit | abnormal |   0.00
absent     | present | present | nonsmoker | false | absent     | no_visit | normal   |   0.00
absent     | present | present | nonsmoker | false | absent     | visit    | abnormal |   0.00
absent     | present | present | nonsmoker | false | absent     | visit    | normal   |   0.00
absent     | present | present | nonsmoker | false | present    | no_visit | abnormal |   0.00
absent     | present | present | nonsmoker | false | present    | no_visit | normal   |   0.00
absent     | present | present | nonsmoker | false | present    | visit    | abnormal |   0.00
absent     | present | present | nonsmoker | false | present    | visit    | normal   |   0.00
absent     | present | present | nonsmoker | true  | absent     | no_visit | abnormal |   0.00
absent     | present | present | nonsmoker | true  | absent     | no_visit | normal   |   0.00
absent     | present | present | nonsmoker | true  | absent     | visit    | abnormal |   0.00
absent     | present | present | nonsmoker | true  | absent     | visit    | normal   |   0.00
absent     | present | present | nonsmoker | true  | present    | no_visit | abnormal |   0.00
absent     | present | present | nonsmoker | true  | present    | no_visit | normal   |   0.00
absent     | present | present | nonsmoker | true  | present    | visit    | abnormal |   0.00
absent     | present | present | nonsmoker | true  | present    | visit    | normal   |   0.00
absent     | present | present | smoker    | false | absent     | no_visit | abnormal |   0.00
absent     | present | present | smoker    | false | absent     | no_visit | normal   |   0.00
absent     | present | present | smoker    | false | absent     | visit    | abnormal |   0.00
absent     | present | present | smoker    | false | absent     | visit    | normal   |   0.00
absent     | present | present | smoker    | false | present    | no_visit | abnormal |   0.00
absent     | present | present | smoker    | false | present    | no_visit | normal   |   0.00
absent     | present | present | smoker    | false | present    | visit    | abnormal |   0.00
absent     | present | present | smoker    | false | present    | visit    | normal   |   0.00
absent     | present | present | smoker    | true  | absent     | no_visit | abnormal |   0.01
absent     | present | present | smoker    | true  | absent     | no_visit | normal   |   0.00
absent     | present | present | smoker    | true  | absent     | visit    | abnormal |   0.00
absent     | present | present | smoker    | true  | absent     | visit    | normal   |   0.00
absent     | present | present | smoker    | true  | present    | no_visit | abnormal |   0.00
absent     | present | present | smoker    | true  | present    | no_visit | normal   |   0.00
absent     | present | present | smoker    | true  | present    | visit    | abnormal |   0.00
absent     | present | present | smoker    | true  | present    | visit    | normal   |   0.00
present    | absent  | absent  | nonsmoker | false | absent     | no_visit | abnormal |   0.00
present    | absent  | absent  | nonsmoker | false | absent     | no_visit | normal   |   0.03
present    | absent  | absent  | nonsmoker | false | absent     | visit    | abnormal |   0.00
present    | absent  | absent  | nonsmoker | false | absent     | visit    | normal   |   0.00
present    | absent  | absent  | nonsmoker | false | present    | no_visit | abnormal |   0.00
present    | absent  | absent  | nonsmoker | false | present    | no_visit | normal   |   0.00
present    | absent  | absent  | nonsmoker | false | present    | visit    | abnormal |   0.00
present    | absent  | absent  | nonsmoker | false | present    | visit    | normal   |   0.00
present    | absent  | absent  | nonsmoker | true  | absent     | no_visit | abnormal |   0.00
present    | absent  | absent  | nonsmoker | true  | absent     | no_visit | normal   |   0.00
present    | absent  | absent  | nonsmoker | true  | absent     | visit    | abnormal |   0.00
present    | absent  | absent  | nonsmoker | true  | absent     | visit    | normal   |   0.00
present    | absent  | absent  | nonsmoker | true  | present    | no_visit | abnormal |   0.00
present    | absent  | absent  | nonsmoker | true  | present    | no_visit | normal   |   0.00
present    | absent  | absent  | nonsmoker | true  | present    | visit    | abnormal |   0.00
present    | absent  | absent  | nonsmoker | true  | present      | visit    | normal   |   0.00
present    | absent  | absent  | smoker    | false | absent       | no_visit | abnormal |   0.00
present    | absent  | absent  | smoker    | false | absent       | no_visit | normal   |   0.05
present    | absent  | absent  | smoker    | false | absent       | visit    | abnormal |   0.00
present    | absent  | absent  | smoker    | false | absent       | visit    | normal   |   0.00
present    | absent  | absent  | smoker    | false | present      | no_visit | abnormal |   0.00
present    | absent  | absent  | smoker    | false | present      | no_visit | normal   |   0.00
present    | absent  | absent  | smoker    | false | present      | visit    | abnormal |   0.00
present    | absent  | absent  | smoker    | false | present      | visit    | normal   |   0.00
present    | absent  | absent  | smoker    | true  | absent       | no_visit | abnormal |   0.00
present    | absent  | absent  | smoker    | true  | absent       | no_visit | normal   |   0.00
present    | absent  | absent  | smoker    | true  | absent       | visit    | abnormal |   0.00
present    | absent  | absent  | smoker    | true  | absent       | visit    | normal   |   0.00
present    | absent  | absent  | smoker    | true  | present      | no_visit | abnormal |   0.00
present    | absent  | absent  | smoker    | true  | present      | no_visit | normal   |   0.00
present    | absent  | absent  | smoker    | true  | present      | visit    | abnormal |   0.00
present    | absent  | absent  | smoker    | true  | present      | visit    | normal   |   0.00
present    | absent  | present | nonsmoker | false | absent       | no_visit | abnormal |   0.01
present    | absent  | present | nonsmoker | false | absent       | no_visit | normal   |   0.11
present    | absent  | present | nonsmoker | false | absent       | visit    | abnormal |   0.00
present    | absent  | present | nonsmoker | false | absent       | visit    | normal   |   0.00
present    | absent  | present | nonsmoker | false | present      | no_visit | abnormal |   0.00
present    | absent  | present | nonsmoker | false | present      | no_visit | normal   |   0.00
present    | absent  | present | nonsmoker | false | present      | visit    | abnormal |   0.00
present    | absent  | present | nonsmoker | false | present      | visit    | normal   |   0.00
present    | absent  | present | nonsmoker | true  | absent       | no_visit | abnormal |   0.00
present    | absent  | present | nonsmoker | true  | absent       | no_visit | normal   |   0.00
present    | absent  | present | nonsmoker | true  | absent       | visit    | abnormal |   0.00
present    | absent  | present | nonsmoker | true  | absent       | visit    | normal   |   0.00
present    | absent  | present | nonsmoker | true  | present      | no_visit | abnormal |   0.00
present    | absent  | present | nonsmoker | true  | present      | no_visit | normal   |   0.00
present    | absent  | present | nonsmoker | true  | present      | visit    | abnormal |   0.00
present    | absent  | present | nonsmoker | true  | present      | visit    | normal   |   0.00
present    | absent  | present | smoker    | false | absent       | no_visit | abnormal |   0.01
present    | absent  | present | smoker    | false | absent       | no_visit | normal   |   0.20
present    | absent  | present | smoker    | false | absent       | visit    | abnormal |   0.00
present    | absent  | present | smoker    | false | absent       | visit    | normal   |   0.00
present    | absent  | present | smoker    | false | present      | no_visit | abnormal |   0.00
present    | absent  | present | smoker    | false | present      | no_visit | normal   |   0.00
present    | absent  | present | smoker    | false | present      | visit    | abnormal |   0.00
present    | absent  | present | smoker    | false | present      | visit    | normal   |   0.00
present    | absent  | present | smoker    | true  | absent       | no_visit | abnormal |   0.00
present    | absent  | present | smoker    | true  | absent       | no_visit | normal   |   0.00
present    | absent  | present | smoker    | true  | absent       | visit    | abnormal |   0.00
present    | absent  | present | smoker    | true  | absent       | visit    | normal   |   0.00
present    | absent  | present | smoker    | true  | present      | no_visit | abnormal |   0.00
present    | absent  | present | smoker    | true  | present      | no_visit | normal   |   0.00
present    | absent  | present | smoker    | true  | present      | visit    | abnormal |   0.00
present    | absent  | present | smoker    | true  | present      | visit    | normal   |   0.00
present    | present | absent  | nonsmoker | false | absent       | no_visit | abnormal |   0.00
present    | present | absent  | nonsmoker | false | absent       | no_visit | normal   |   0.00
present    | present | absent  | nonsmoker | false | absent       | visit    | abnormal |   0.00
present    | present | absent  | nonsmoker | false | absent       | visit    | normal   |   0.00
present    | present | absent  | nonsmoker | false | present      | no_visit | abnormal |   0.00
present    | present | absent  | nonsmoker | false | present      | no_visit | normal   |   0.00
present    | present | absent  | nonsmoker | false | present      | visit    | abnormal |   0.00
present    | present | absent  | nonsmoker | false | present      | visit    | normal   |   0.00
present    | present | absent  | nonsmoker | true  | absent       | no_visit | abnormal |   0.00
present    | present | absent  | nonsmoker | true  | absent       | no_visit | normal   |   0.00
present    | present | absent  | nonsmoker | true  | absent       | visit    | abnormal |   0.00
present    | present | absent  | nonsmoker | true  | absent       | visit    | normal   |   0.00
present    | present | absent  | nonsmoker | true  | present      | no_visit | abnormal |   0.00
present    | present | absent  | nonsmoker | true  | present      | no_visit | normal   |   0.00
present    | present | absent  | nonsmoker | true  | present      | visit    | abnormal |   0.00
present    | present | absent  | nonsmoker | true  | present      | visit    | normal   |   0.00
present    | present | absent  | smoker    | false | absent       | no_visit | abnormal |   0.00
present    | present | absent  | smoker    | false | absent       | no_visit | normal   |   0.00
present    | present | absent  | smoker    | false | absent       | visit    | abnormal |   0.00
present    | present | absent  | smoker    | false | absent       | visit    | normal   |   0.00
present    | present | absent  | smoker    | false | present      | no_visit | abnormal |   0.00
present    | present | absent  | smoker    | false | present      | no_visit | normal   |   0.00
present    | present | absent  | smoker    | false | present      | visit    | abnormal |   0.00
present    | present | absent  | smoker    | false | present      | visit    | normal   |   0.00
present    | present | absent  | smoker    | true  | absent       | no_visit | abnormal |   0.00
present    | present | absent  | smoker    | true  | absent       | no_visit | normal   |   0.00
present    | present | absent  | smoker    | true  | absent       | visit    | abnormal |   0.00
present    | present | absent  | smoker    | true  | absent       | visit    | normal   |   0.00
present    | present | absent  | smoker    | true  | present      | no_visit | abnormal |   0.00
present    | present | absent  | smoker    | true  | present      | no_visit | normal   |   0.00
present    | present | absent  | smoker    | true  | present      | visit    | abnormal |   0.00
present    | present | absent  | smoker    | true  | present      | visit    | normal   |   0.00
present    | present | present | nonsmoker | false | absent       | no_visit | abnormal |   0.00
present    | present | present | nonsmoker | false | absent       | no_visit | normal   |   0.00
present    | present | present | nonsmoker | false | absent       | visit    | abnormal |   0.00
present    | present | present | nonsmoker | false | absent       | visit    | normal   |   0.00
present    | present | present | nonsmoker | false | present      | no_visit | abnormal |   0.00
present    | present | present | nonsmoker | false | present      | no_visit | normal   |   0.00
present    | present | present | nonsmoker | false | present      | visit    | abnormal |   0.00
present    | present | present | nonsmoker | false | present      | visit    | normal   |   0.00
present    | present | present | nonsmoker | true  | absent       | no_visit | abnormal |   0.00
present    | present | present | nonsmoker | true  | absent       | no_visit | normal   |   0.00
present    | present | present | nonsmoker | true   | absent       | visit     | abnormal |   0.00
present    | present | present | nonsmoker | true   | absent       | visit     | normal   |   0.00
present    | present | present | nonsmoker | true   | present      | no_visit  | abnormal |   0.00
present    | present | present | nonsmoker | true   | present      | no_visit  | normal   |   0.00
present    | present | present | nonsmoker | true   | present      | visit     | abnormal |   0.00
present    | present | present | nonsmoker | true   | present      | visit     | normal   |   0.00
present    | present | present | smoker    | false  | absent       | no_visit  | abnormal |   0.00
present    | present | present | smoker    | false  | absent       | no_visit  | normal   |   0.00
present    | present | present | smoker    | false  | absent       | visit     | abnormal |   0.00
present    | present | present | smoker    | false  | absent       | visit     | normal   |   0.00
present    | present | present | smoker    | false  | present      | no_visit  | abnormal |   0.00
present    | present | present | smoker    | false  | present      | no_visit  | normal   |   0.00
present    | present | present | smoker    | false  | present      | visit     | abnormal |   0.00
present    | present | present | smoker    | false  | present      | visit     | normal   |   0.00
present    | present | present | smoker    | true   | absent       | no_visit  | abnormal |   0.03
present    | present | present | smoker    | true   | absent       | no_visit  | normal   |   0.00
present    | present | present | smoker    | true   | absent       | visit     | abnormal |   0.00
present    | present | present | smoker    | true   | absent       | visit     | normal   |   0.00
present    | present | present | smoker    | true   | present      | no_visit  | abnormal |   0.00
present    | present | present | smoker    | true   | present      | no_visit  | normal   |   0.00
present    | present | present | smoker    | true   | present      | visit     | abnormal |   0.00
present    | present | present | smoker    | true   | present      | visit     | normal   |   0.00
```

The default precision of two decimal places causes a lot of rounding errors in the presentation. To change this pull the `Parameters` module into your namespace and alter the value of its `precision` variable.

```
>>> import gPy.Parameters
>>> gPy.Parameters.precision = 4
```

If you were to print out `joint` again (I won't since it takes up so much space!) you would get some more numbers after the decimal point.

Chapter 3

Hypergraphs

In `gPy` *hypergraphs*, rather than graphs, are the central structure; although the latter still play an important role. A hypergraph ($\mathcal{H}$) is simply a collection of subsets of a finite set (H). These subsets ($h \in \mathcal{H}$) are known as *hyperedges*. H is called the *base set*. The elements of H are known as *vertices*. In `gPy`, hypergraphs are restricted so that $H = \bigcup_{h\in\mathcal{H}} h$: every element of the base set is contained in at least one hyperedge. Table 3.1 gives some example hypergraphs (don't worry about the final 4 columns for now).

3.1 Constructing hypergraphs

In `gPy`, hypergraphs are object of the class `gPy.Structures.HyperGraph` (or one of its subclasses). To construct a hypergraph it suffices to send the desired hyperedges to the `HyperGraph` constructor method:

```
>>> from gPy.Structures import HyperGraph
>>> hg1 = HyperGraph(['AB','BC','CD','DA'])
>>> hg2 = HyperGraph(([],['A'],['A','B','C']))
>>> print hg1
( {B, C}, {A, D}, {A, B}, {C, D} )
>>> print hg2
( {A, B, C}, {}, {A} )
```

$\mathcal{H}$	Simple?	Reduced?	Graphical?	Decomposable?
$(\{A\},\{A\},\{A,B\})$	N	N	Y	Y
$\{\emptyset,\{A\},\{A,B\}\}$	Y	N	Y	Y
$\{\{A,B\},\{B,C\},\{A,C\}\}$	Y	Y	N	N
$\{\{A,B\},\{B,C\},\{C,D\},\{A,D\}\}$	Y	Y	Y	N
$\{\{A,B\},\{B,C\}\}$	Y	Y	Y	Y

Table 3.1: Classification of some example hypergraphs

Each vertex should be some immutable object (in most real applications it will be a string—the name of some random variable). Each hyperedge is an *iterable* of vertices, this will normally just be a sequence such as a list or tuple, but sets can be used as well. Note that in Python a string is seen as a sequence of characters. This was used in the construction of `hg1` above.

Empty hypergraphs are permissible, in fact the empty tuple is the default collection of hyperedges.

```
>>> e = HyperGraph()
>>> print e
(  )
```

Empty hypergraphs are useful as 'initial' hypergraphs to which hyperedges can be added later.

3.2 Internal representation of hypergraphs

Internally, each hyperedge is stored as a `frozenset` of vertices. Each hypergraph has a private attribute `_hyperedges` which is just the `set` of these hyperedges which constitute the hypergraph. For simple hypergraphs—of which more later—this attribute contains all the information required to define the hypergraph. However, it is useful to also maintain a mapping from each vertex to the the set of hyperedges which contain it–its *star*. (This is essentially the *dual hypergraph* [1].) This mapping (a Python dictionary) is the private attribute `_star`. Here is the internal representation of the `hg1` hypergraph given earlier (with added fake line breaks to make the output readable):

```
>>> for att, val in vars(hg1).items():
...  print att
...  print val
...  print
...
_hyperedges
set([frozenset(['C', 'B']), frozenset(['A', 'D']), <FAKE_NL>
frozenset(['A', 'B']), frozenset(['C', 'D'])])

_star
{'A': set([frozenset(['A', 'D']), frozenset(['A', 'B'])]), <FAKE_NL>
'C': set([frozenset(['C', 'B']), frozenset(['C', 'D'])]), <FAKE_NL>
'B': set([frozenset(['C', 'B']), frozenset(['A', 'B'])]), <FAKE_NL>
'D': set([frozenset(['A', 'D']), frozenset(['C', 'D'])])}
```

3.3 Altering hypergraphs

Hypergraphs are *mutable* objects and so can be altered. The two basic operators are: `add_hyperedge` and `remove_hyperedge` which work as follows:

```
>>> print hg1
( {B, C}, {A, D}, {A, B}, {C, D} )
>>> hg1.add_hyperedge('ABC')
>>> print hg1
( {A, B, C}, {B, C}, {A, D}, {A, B}, {C, D} )
>>> hg1.remove_hyperedge(('A','B'))
>>> print hg1
( {A, B, C}, {B, C}, {A, D}, {C, D} )
```

Note the flexibility with which hyperedges are specified. Any sequence will do (in fact, any iterable).

3.4 Copying hypergraphs

As well as constructing hypergraphs directly from collections of hyperedges, new hypergraphs can be made by copying existing hypergraphs. The copy is a 'deep' copy: it is completely independent from the original, any alterations done to one will have no effect on the other.

```
>>> hg = HyperGraph(['AB','BC','CD','DA'])
>>> hg_cp = hg.copy()
>>> hg.add_hyperedge('EF')
>>> hg_cp.remove_hyperedge('AB')
>>> print hg
( {E, F}, {B, C}, {A, D}, {A, B}, {C, D} )
>>> print hg_cp
( {C, D}, {B, C}, {A, D} )
```

3.5 Making graphs from hypergraphs

Each hypergraph $\mathcal{H}$ has an associated (undirected) graph $\mathcal{H}_{[2]}$, called its 2-section. $\mathcal{H}_{[2]}$ has the same vertices as $\mathcal{H}$ and two vertices are connected in $\mathcal{H}_{[2]}$ if and only if they are both elements of some hyperedge in $\mathcal{H}$. 2-sections are generated using the `two_section` method of `gPy.Structures.HyperGraph`:

```
>>> print hg1
( {A, B, C}, {B, C}, {A, D}, {C, D} )
>>> print hg1.two_section()
Vertices:
['A', 'B', 'C', 'D']
Lines:
```

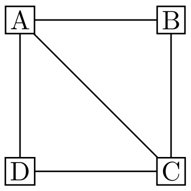

Figure 3.1: The graph of the hypergraph $\{\{A,B,C\},\{B,C\},\{A,D\},\{C,D\}\}$

```
A - B
A - C
A - D
B - C
C - D
```

Fig 3.1 provides an example of a two-section graph generated from a hypergraph.

3.6 Hypergraph subclasses

The `gPy.Structures.HyperGraph` class is for general hypergraphs. `gPy` also defines subclasses of `gPy.Structures.HyperGraph` for hypergraphs with certain properties. The hierarchy is given in Fig 3.2.

A *simple* hypergraph is one with no repeated hyperedges. A *reduced* hypergraph contains no hyperedge contained within another. A reduced hypergraph is thus always simple. To understand what a *graphical* hypergraph is we need to make reference to its graph. Recall that each hypergraph $\mathcal{H}$ has an associated graph $\mathcal{H}_{[2]}$ as explained in Section 3.5. It is also the case that each undirected graph $\mathcal{G}$ has an associated hypergraph $\mathcal{C}(\mathcal{G})$ defined in terms of its *cliques*. A clique in a graph is a maximally complete subset of vertices of the graph. Each vertex in a clique is connected to all other vertices in the clique (that's completeness) and no vertex can be added to the clique without it becoming incomplete (that's maximal completeness). Since each clique is just a set of vertices, it follows that the set of all cliques of a graph is a hypergraph, the *clique hypergraph* of the graph.

To generate the clique hypergraph `gPy` provides the `gPy.Structures.UGraph` method `hypergraph`. Here we generate the hypergraph of the graph displayed in Fig 3.1.

```
>>> g1 = hg1.two_section()
>>> print g1
```

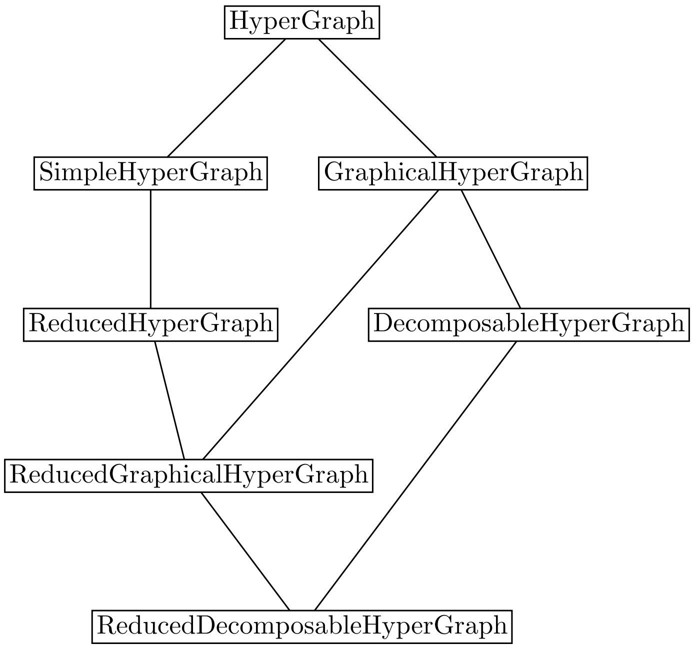

Figure 3.2: Hypergraph hierarchy

```
Vertices:
['A', 'B', 'C', 'D']
Lines:
A - B
A - C
A - D
B - C
C - D

>>> print g1.hypergraph()
{ {A, B, C}, {A, C, D} }
```

It is interesting to examine the relationship between a hypergraph $\mathcal{H}$:

```
>>> print hg1
( {A, B, C}, {B, C}, {A, D}, {C, D} )
```

and $\mathcal{C}(\mathcal{H}_{[2]})$, the clique hypergraph of its two-section:

```
>>> print hg1.two_section().hypergraph()
{ {A, B, C}, {A, C, D} }
```

It is not difficult to prove that for any hypergraph $\mathcal{H}$, every hyperedge in $\mathcal{H}$ is contained within some hyperedge in $\mathcal{C}(\mathcal{H}_{[2]})$. However, the reverse does not always follow. In the example above the hyperedge $\{A, C, D\}$ of $\mathcal{C}(\mathcal{H}_{[2]})$ is not contained in any hyperedge of $\mathcal{H}$. This means that $\mathcal{H}$ is *not graphical*. This leads us finally to the definition of a graphical hypergraph: a hypergraph $\mathcal{H}$ is *graphical* iff every hyperedge of $\mathcal{C}(\mathcal{H}_{[2]})$ is contained in some hyperedge of $\mathcal{H}$. It follows that if a hypergraph $\mathcal{H}$ is both graphical and reduced then $\mathcal{H} = \mathcal{C}(\mathcal{H}_{[2]})$. A hypergraph is *decomposable* iff it is graphical and its two-section is *triangulated*. A graph is triangulated if every cycle of length 4 or more has a short-cut (a *chord*). Triangulated graphs are also called *chordal graphs*.

3.7 Constructing hypergraphs with specific properties

gPy provides the classes `HyperGraph`,`SimpleHyperGraph`, `ReducedHyperGraph`, `GraphicalHyperGraph`, `ReducedGraphicalHyperGraph`, `DecomposableHyperGraph` and `ReducedDecomposableHyperGraph` for the various types of hypergraph. Recall Fig 3.2 which gives the hierarchical relationships between these classes. In Section 3.1 the method of constructing `HyperGraph` objects was described. To construct objects of any of `HyperGraph`'s subclasses, one option is to send a `HyperGraph` to the class constructor, if this object meets the conditions of the subclass (i.e. is reduced, graphical, decomposable, etc) then an object of the required class is returned, if not an exception is raised. The returned object will share its attributes with the inputted object. Here's some examples of successful attempts to create various objects:

```
>>> from gPy.Structures import *
>>> sg = SimpleHyperGraph(HyperGraph(['ABC', 'BCD']))
>>> print sg
{ {B, C, D}, {A, B, C} }
>>> dg = DecomposableHyperGraph(HyperGraph(['ABC', 'BCD']))
>>> gg = GraphicalHyperGraph(HyperGraph(['AB', 'BC', 'CD', 'AD']))
>>> rg = ReducedHyperGraph(HyperGraph(['AB', 'BC', 'CA']))
```

Note that the simple hypergraph sg is printed out using '{' and '}' as delimiters to make evident that that it is a set of hyperedges. Generally, a hypergraph need not be a set of hyperedges due to the possible existence of repeated hyperedges.

Here are some failed attempts to create particular sorts of hypergraphs:

```
>>> print hg1
( {A, B, C}, {B, C}, {A, D}, {C, D} )
>>> rhg = ReducedHyperGraph(hg1)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 1871, in __init__
    raise RedundancyError("%s is not reduced" % hypergraph)
gPy.Structures.RedundancyError: ( {A, B, C}, {B, C}, {A, D}, {C, D} ) is not reduced
>>> ghg = GraphicalHyperGraph(hg1)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 1959, in __init__
    raise GraphicalityError("%s is not graphical" % hypergraph)
gPy.Structures.GraphicalityError: ( {A, B, C}, {B, C}, {A, D}, {C, D} ) is not graphical
>>> dhg = DecomposableHyperGraph(hg1)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 2049, in __init__
    JoinForest(hypergraph,modify,trace,elimination_order)
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 2159, in __init__
    hypergraph._uforest = hypergraph.join_forest()
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 814, in join_forest
    raise DecomposabilityError("%s is not decomposable" % self)
gPy.Structures.DecomposabilityError: ( {A, B, C}, {B, C}, {A, D}, {C, D} ) is not decomposal
```

The constructor for the general hypergraph class HyperGraph can also be used in this way: being sent a hypergraph as input rather than a collection of edges:

```
>>> dg = DecomposableHyperGraph(HyperGraph(['ABC', 'BCD']))
>>> type(dg)
<class 'gPy.Structures.DecomposableHyperGraph'>
>>> new_hg = HyperGraph(dg)
>>> type(new_hg)
<class 'gPy.Structures.HyperGraph'>
```

This approach would typically be used if we want to 'forget' that a particular hypergraph has certain properties. For example, suppose we wanted to add a redundant hyperedge to a hypergraph of class `ReducedHyperGraph` thus rendering it no longer reduced. Doing so would cause an exception, so it is necessary to create a `HyperGraph` object first and then add the redundant hyperedge:

```
>>> rg = ReducedHyperGraph(HyperGraph(['AB', 'BC', 'CA']))
>>> rg.add_hyperedge('A')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 689, in add_hyperedge
    raise RedundancyError("%s would make %s no longer reduced" % (hyperedge,self))
gPy.Structures.RedundancyError: Adding {A} would make { {B, C}, {A, C}, {A, B} } no longer r
>>> hg = HyperGraph(rg)
>>> hg.add_hyperedge('A')
>>> print hg
( {B, C}, {A, C}, {A, B}, {A} )
```

As a convenience it is also possible to create any type of hypergraph directly from hyperedges, exactly as described for the `HyperGraph` class in Section 3.1:

```
>>> dg1 = DecomposableHyperGraph(['ABC', 'BCD'])
>>> gg1 = GraphicalHyperGraph(['AB', 'BC', 'CD', 'AD'])
>>> rg1 = ReducedHyperGraph(['AB', 'BC', 'CA'])
```

When called in this way a temporary internal `HyperGraph` is created from the hyperedges, and the object is constructed from this temporary `HyperGraph` exactly as it had been supplied as an argument.

3.7.1 Potential pitfalls in creating hypergraphs

Recall that a new hypergraph created from an old one has identical attributes to the original hypergraph. This can lead a hypergraph to get into an illegal state. Continuing the example immediately above:

```
>>> rg._hyperedges is hg._hyperedges
True
>>> print rg
{ {B, C}, {A, C}, {A, B}, {A} }
>>> type(rg)
<class 'gPy.Structures.ReducedHyperGraph'>
```

Because `rg` and `hg` share the same `_hyperedges` attribute, adding the redundant hyperedge to `hg` also adds it to `rg` even though the latter is of class `ReducedHyperGraph`. This behaviour is allowed in `gPy` since it is sometimes useful to have distinct hypergraph objects sharing the same attributes. Also, it is very easy to avoid the problem of illegal states: construct hypergraphs using unnamed input hypergraphs, in which case the input hypergraph will be immediately garbage collected. This approach was taken in many of the examples given above. Alternatively, a named input hypergraph can be used if it not used afterwards (and so will be garbage collected at some point), or it can even be explicitly deleted (although this would rarely be necessary). If you want to create a new hypergraph from an existing one, and wish the two to have independent existences, just use a copy—which will be a deep copy—of the existing hypergraph to create the new one:

```
>>> hg = HyperGraph(('AB','BC','CD'))
>>> rhg = ReducedHyperGraph(hg.copy())
>>> hg.add_hyperedge('A')
>>> print hg
( {B, C}, {A, B}, {A}, {C, D} )
>>> print rhg
{ {B, C}, {A, B}, {C, D} }
```

3.7.2 Creating hypergraphs without checks

In some cases you may know that a particular hypergraph has the properties required for a hypergraph class 'below' it in the hierarchy (Fig 3.2) and wish to avoid the cost of pointlessly checking that it meets the necessary conditions. In this case, an object of the relevant class can be created using the optional `check` argument set to `False` (the default it `True`). This facility should only be used if you are absolutely sure that no checking is required: otherwise you can, for example, create a `ReducedHyperGraph` object that is not reduced!

```
>>> print hg
( {B, C}, {A, B}, {A}, {C, D} )
>>> r = ReducedHyperGraph(hg.copy())
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 678, in __init__
    if hypergraph is None:
gPy.Structures.RedundancyError: ( {B, C}, {A, B}, {A}, {C, D} ) is not reduced
>>> r = ReducedHyperGraph(hg.copy(),check=False)
>>> print r
{ {B, C}, {A, B}, {A}, {C, D} }
```

3.7.3 Creating hypergraphs by modifying existing hypergraphs

It is often useful to create a hypergraph meeting certain conditions from a hypergraph which does not meet these conditions. For example, we can create a reduced hypergraph from an arbitrary hypergraph by simply deleting redundant hyperedges. Graphical and decomposable hypergraphs can be created from arbitrary hypergraphs by merging hyperedges. To do this set the optional `modify` constructor argument to `True` (default is `False`). Firstly, consider doing this for ReducedHyperGraph objects since in this case there is one obvious choice for modification—deleting redundant hyperedges:

```
>>> print hg
( {B, C}, {A, B}, {A}, {C, D} )
>>> rhg = ReducedHyperGraph(hg.copy(),modify=True)
>>> print rhg
{ {B, C}, {A, B}, {C, D} }
```

At time of writing gPy provides no way of creating GraphicalHyperGraph and ReducedGraphicalHyperGraph objects from hypergraphs which fail to meet the necessary conditions. However, it is possible to create a DecomposableHyperGraph from a non-decomposable hypergraph.

Creating decomposable hypergraphs

To create a decomposable hypergraph—a DecomposableHyperGraph object—from an existing hypergraph there are a number of options. If the existing hypergraph is decomposable then calls such as:

```
>>> dg = DecomposableHyperGraph(HyperGraph(['ABC', 'BCD']))
>>> dg = DecomposableHyperGraph(['ABC', 'BCD'])
>>> dg = DecomposableHyperGraph(['ABC', 'BCD'],check=False)
```

suffice as previously explained.

If it is not safe to assume that the input hypergraph is decomposable then it can be modified to become decomposable by having some of its hyperedges merged. Each *elimination ordering* of the vertices of an arbitrary hypergraph determines a decomposable hypergraph, so one option for making decomposable hypergraphs is to supply such an ordering:

```
>>> hg = HyperGraph(['A','AB','BC','CD','AD'])
>>> dg = DecomposableHyperGraph(hg,modify=True,elimination_order='ABCD')
>>> print dg
( {B, C, D}, {A, B, D} )
>>> print hg
( {B, C, D}, {A, B, D} )
>>> print type(dg)
<class 'gPy.Structures.DecomposableHyperGraph'>
>>> print type(hg)
<class 'gPy.Structures.HyperGraph'>
```

Note that hg and dg are now identical except for their class. Had we wished to keep the original hg intact we would have sent hg.copy() to the DecomposableHyperGraph constructor instead.

Some elimination orders are better than others in the sense that they do not create big hyperedges in the created decomposable hypergraph:

```
>>> hg = HyperGraph(['AB','AC','AD','AE'])
>>> print hg
( {A, E}, {A, D}, {A, C}, {A, B} )
>>> dg_bad = DecomposableHyperGraph(hg.copy(),modify=True,elimination_order='ABCDE')
>>> dg_good = DecomposableHyperGraph(hg.copy(),modify=True,elimination_order='BCDEA')
>>> print dg_bad
( {A, B, C, D, E} )
>>> print dg_good
( {A, E}, {A, D}, {A, C}, {A, B} )
>>> dg_lessbad = DecomposableHyperGraph(hg.copy(),modify=True,elimination_order='BCADE')
>>> print dg_lessbad
( {A, C}, {A, B}, {A, D, E} )
```

`hg` here is in fact already decomposable, but by choosing to eliminate `A` first we create `dg_bad` in which all hyperedges of `hg` have been merged into one big hyperedge. The elimination order where `A` is left to last avoids the need to merge any hyperedges (`dg_good`) whereas `dg_lessbad` does some unnecessary merging, but less than `dg_bad`.

Naturally, it would be nice to automatically find good elimination orders. Unfortunately this is a NP-hard problem, but we can still find reasonable strategies. In `gPy`, if no elimination order is supplied restricted maximum cardinality search on hypergraphs [3] is used to find one. Maximum cardinality search has the nice property that if the input hypergraph is already decomposable an ordering is found which does not create any new hyperedges:

```
>>> dg = DecomposableHyperGraph(hg.copy(),modify=True)
>>> print dg
( {A, E}, {A, D}, {A, C}, {A, B} )
```

If the input hypergraph is not decomposable then the ordering is usually reasonably good, but not always. Here's an example of a bad ordering (`'ABCDE'`) creating a hypergraph with big hyperedges and the internally generated maximum cardinality search ordering doing a better job:

```
>>> hg2 = HyperGraph(['AB','BC','CD','DA','AE'])
>>> dg2a = DecomposableHyperGraph(hg2.copy(),modify=True,elimination_order='ABCDE')
>>> dg2b = DecomposableHyperGraph(hg2.copy(),modify=True)
>>> print dg2a
( {B, C, D, E}, {A, B, D, E} )
>>> print dg2b
( {B, C, D}, {A, B, D}, {A, E} )
```

3.8 Join Forests

To each decomposable hypergraph there is one or more associated *join forests*. A join forest $\mathcal{F}$ for a decomposable hypergraph $\mathcal{H}$, is an undirected graph whose vertices are the hyperedges of $\mathcal{H}$with the following two properties:

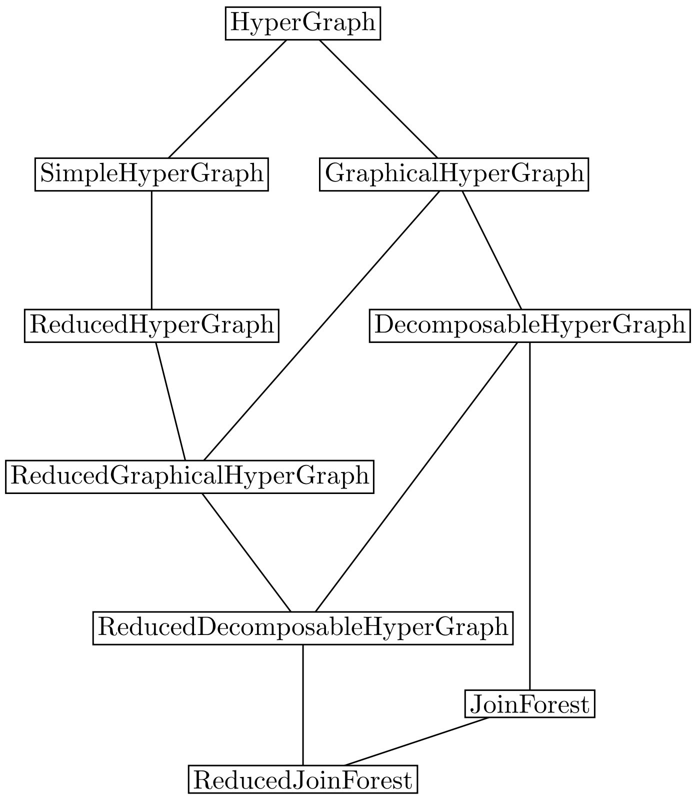

Figure 3.3: Hypergraph hierarchy

1. The graph is a `forest`: it is the disjoint union of one or more `trees`. An (undirected) tree is a graph which contains no cycles.
2. The join forest $\mathcal{F}$ obeys the *join property*. If hyperedges h_1 and h_2 are in the same tree in $\mathcal{F}$, then for any hyperedge h_3 on the (unique) path between h_1 and h_2 we have: $h_1 \cap h_2 \subseteq h_3$.

Join forests are also often referred to as `junction forests`, for example in [2]. Also many presentations make the assumption that there is only one tree in the forest and so talk about *join trees* or *junction trees*.

In `gPy`, join forests are created similarly to `DecomposableHyperGraph` objects with one differences: the `JoinForest` constructor does not accept a `check` argument since the creation of a join forest inevitably checks that the input hypergraph is decomposable. The `JoinForest` class is a direct subclass of the `DecomposableHyperGraph` class, see Fig 3.3 for the full hierarchy of hypergraphs. `JoinForest` objects have an additional `_uforest` attribute which contains the join forest itself. The join forest is a `gPy.Structures.UForest` object.

Here's two examples of creating `JoinForest` objects:

```
>>> hg2 = HyperGraph(['AB','BC','CD','DA','AE'])
>>> jf2a = JoinForest(hg2.copy(),modify=True,elimination_order='ABCDE')
>>> print jf2a
{ {B, C, D, E}, {A, B, D, E} }
Vertices:
[{B, C, D, E}, {A, B, D, E}]
Lines:
{A, B, D, E} - {B, C, D, E}

>>> jf2b = JoinForest(hg2.copy(),modify=True)
>>> print jf2b
{ {B, C, D}, {A, B, D}, {A, E} }
Vertices:
[{B, C, D}, {A, B, D}, {A, E}]
Lines:
{A, B, D} - {B, C, D}
{A, E} - {A, B, D}
```

TRACE STUFF!!!!!

Chapter 4

Graphs

Chapter 5

Factors

5.1 Accessing data in factors

Data in a factor can be accessed using the usual Python indexing syntax: `[]`. There are a three permissible forms of index:

Dictionary If a dictionary is supplied as an index, the keys of the dictionary should be the variables in the factor. Associated with each variable key should be a single value for that variable. The single data value associated with the joint instantiation thus specified is returned.

Sequence If a sequence is supplied it should be a sequence of variable values, ordered according to the lexicographical ordering of the variable names. This specifies a joint instantiation corresponding to a row of the printed representation of a factor, and the corresponding data value is returned.

Anything else Any index which is not a dictionary or a sequence is viewed as an index for the internal `_data` attribute, a list which holds the data.

Here are some examples of using these various indexing options to yank out data values:

```
>>> print factor

Bronchitis | Dyspnea | TbOrCa |
---------- | ------- | ------ | ----
absent     | absent  | false  |   0.90
absent     | absent  | true   |   0.30
absent     | present | false  |   0.10
absent     | present | true   |   0.70
present    | absent  | false  |   0.20
present    | absent  | true   |   0.10
present    | present | false  |   0.80
present    | present | true   |   0.90
```

```
>>> factor[2]
0.10000000000000001
>>> factor['absent','absent','true']
0.29999999999999999
>>> factor[{'Dyspnea':'absent','Bronchitis':'absent','TbOrCa':'true'}]
0.29999999999999999
>>> factor[2:4]
[0.10000000000000001, 0.69999999999999996]
```

One can also set data values in the same way.

```
>>> factor[{'Bronchitis':'absent','TbOrCa':'false','Dyspnea':'absent'}] = 0.7
>>> print factor

Bronchitis | Dyspnea | TbOrCa |
---------- | ------- | ------ | ----
absent     | absent  | false  |   0.70
absent     | absent  | true   |   0.30
absent     | present | false  |   0.10
absent     | present | true   |   0.70
present    | absent  | false  |   0.20
present    | absent  | true   |   0.10
present    | present | false  |   0.80
present    | present | true   |   0.90

>>> factor['absent','present','true'] = 1.2
>>> print factor

Bronchitis | Dyspnea | TbOrCa |
---------- | ------- | ------ | ----
absent     | absent  | false  |   0.70
absent     | absent  | true   |   0.30
absent     | present | false  |   0.10
absent     | present | true   |   1.20
present    | absent  | false  |   0.20
present    | absent  | true   |   0.10
present    | present | false  |   0.80
present    | present | true   |   0.90

>>> factor[5:] = 1,2,3
>>> print factor

Bronchitis | Dyspnea | TbOrCa |
---------- | ------- | ------ | ----
absent     | absent  | false  |   0.70
absent     | absent  | true   |   0.30
absent     | present | false  |   0.10
absent     | present | true   |   1.20
present    | absent  | false  |   0.20
present    | absent  | true   |   1.00
present    | present | false  |   2.00
present    | present | true   |   3.00
```

5.2 Conditional probability tables

Conditional probability tables (from now on abbreviated to CPTs) are a special type of factor which define an (in general) different probability distribution for one of factor's variables (the *child* variable) *conditional* on each joint instantiation of the other variables in the factor. These are the factors which parameterise a Bayesian network.

5.2.1 Constructing CPTs

The only way of constructing a CPT is to use an existing factor and specify which variable is to be the child. The simplest case is when the factor already happens to be a valid CPT:

```
>>> print factor

Bronchitis | Dyspnea | TbOrCa |
---------- | ------- | ------ | ----
absent     | absent  | false  |   0.90
absent     | absent  | true   |   0.30
absent     | present | false  |   0.10
absent     | present | true   |   0.70
present    | absent  | false  |   0.20
present    | absent  | true   |   0.10
present    | present | false  |   0.80
present    | present | true   |   0.90

>>> from gPy.Parameters import CPT
>>> cpt = CPT(factor,'Dyspnea')
>>> print cpt

Bronchitis | TbOrCa |      Dyspnea
           |        |   absent  present
---------- | ------ | -----------------
absent     | false  |     0.90     0.10
absent     | true   |     0.30     0.70
present    | false  |     0.20     0.80
present    | true   |     0.10     0.90
```

Note that a factor which is a CPT has a different printed representation to a normal factor. (Internally, the only difference, apart from the class, is the specification of the child variable.)

Potential pitfalls in creating CPTs

After a CPT object has been created, the factor used to create it has identical shared attributes with the CPT, *including a `_child` attribute specifying the child variable.* The only difference is the class of the objects. To avoid any possible confusion a safe bet is to use a copy of a factor to create a CPT:

```
>>> print f

Bronchitis | Dyspnea | TbOrCa |
---------- | ------- | ------ | ----
absent     | absent  | false  |   0.90
absent     | absent  | true   |   0.30
absent     | present | false  |   0.10
absent     | present | true   |   0.70
present    | absent  | false  |   0.20
present    | absent  | true   |   0.10
present    | present | false  |   0.80
present    | present | true   |   0.90

>>> cpt = CPT(f.copy(),'Dyspnea')
```

Options for creating CPTs

Firstly, note that the default behaviour for creating CPTs is *not* to check that the resulting object is a valid CPT. This is because it is often useful to create 'fake' CPTs. To check that the factor used to create a CPT is indeed a CPT, use the `cpt_check` flag:

```
>>> cpt = CPT(f.copy(),'Bronchitis',cpt_check=True)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Parameters.py", line 919, in __init__
    raise CPTError(errmsg)
gPy.Parameters.CPTError:
For child: Bronchitis
For row: [0, 4]
Sum was: 1.10 (should be 1.0)
```

In other cases, it is useful to normalise data values in the input factor so that the factor is forced to become a CPT. To do this use the `cpt_force` flag:

```
>>> cpt = CPT(f.copy(),'Bronchitis',cpt_force=True)
>>> print cpt

Dyspnea | TbOrCa |     Bronchitis
        |        |   absent  present
------- | ------ | -----------------
absent  | false  |     0.82     0.18
absent  | true   |     0.75     0.25
present | false  |     0.11     0.89
present | true   |     0.44     0.56
```

Sometimes this normalisation can produce a division by zero which raises an exception. It is sometimes convenient to permit a 'fake' CPT to be created even in this situation. To do so, use the `allow_dummies` flag:

```
>>> print f

Cancer  | TbOrCa | Tuberculosis |
------- | ------ | ------------ | ----
absent  | false  | absent       |   1.00
absent  | false  | present      |   0.00
absent  | true   | absent       |   0.00
absent  | true   | present      |   1.00
present | false  | absent       |   0.00
present | false  | present      |   0.00
present | true   | absent       |   1.00
present | true   | present      |   1.00

>>> cpt = CPT(f.copy(),'Tuberculosis',cpt_force=True)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Parameters.py", line 911, in __init__
    self._data[indx] /= prob_sum
ZeroDivisionError: float division
>>> cpt = CPT(f.copy(),'Tuberculosis',cpt_force=True,allow_dummies=True)
>>> print cpt

Cancer  | TbOrCa |    Tuberculosis
        |        |   absent  present
------- | ------ | -----------------
absent  | false  |     1.00     0.00
absent  | true   |     0.00     1.00
present | false  |     0.00     0.00
present | true   |     0.50     0.50
```

Note that the row of the CPT where the zero division error has been suppressed has a row of zeroes. This is to flag that some cheating has been going on. Naturally using this row as if actually defined a proper distribution over `Tuberculosis` will cause an error at a later stage.

5.2.2 Extracting data from CPTs

CPTs can be indexed exactly like any other factor is desired. But they also come with an extra indexing possibilities. If the index corresponds to a *row* of the CPT then a single-variable CPT for the child is returned, rather than just data:

```
>>> print cpt

Bronchitis | TbOrCa |      Dyspnea
           |        |   absent  present
---------- | ------ | -----------------
absent     | false  |     0.90     0.10
absent     | true   |     0.30     0.70
present    | false  |     0.20     0.80
present    | true   |     0.10     0.90

>>> print cpt['absent','true']

     Dyspnea
  absent  present
-----------------
    0.30     0.70

>>> print cpt[{'Bronchitis':'absent','TbOrCa':'true'}]

     Dyspnea
  absent  present
-----------------
    0.30     0.70
```

To get at individual numbers is a CPT there are two options: either index exactly as for arbitrary factors:

```
>>> print cpt[{'Bronchitis':'absent','TbOrCa':'true','Dyspnea':'present'}]
0.7
```

or use two indices:

```
>>> print cpt[{'Bronchitis':'absent','TbOrCa':'true'}][['present']]
0.7
```

Note how, in the last example, it was necessary to use `['present']` as an index rather than `'present'`. This is because the index will be interpreted as a sequence.

Iterating over CPTs

gPy provides a number of methods for iterating over the rows of CPTs. For example parent_insts allows you to iterate through the possible joint instantiations of the parents in a CPT:

```
>>> from gPy.Examples import asia
>>> print asia['Dyspnea']

Bronchitis | TbOrCa |      Dyspnea
           |        |   absent  present
---------- | ------ | -----------------
absent     | false  |     0.90     0.10
absent     | true   |     0.30     0.70
present    | false  |     0.20     0.80
present    | true   |     0.10     0.90

>>> for row in asia['Dyspnea'].parent_insts():
...  print row
...
('absent', 'false')
('absent', 'true')
('present', 'false')
('present', 'true')
```

The parent_insts_data method iterates over the data values corresponding to each instantiation of the parents:

```
>>> for data_row in asia['Dyspnea'].parent_insts_data():
...  print data_row
...
[0.90000000000000002, 0.10000000000000001]
[0.29999999999999999, 0.69999999999999996]
[0.20000000000000001, 0.80000000000000004]
[0.10000000000000001, 0.90000000000000002]
```

Both of these methods return iterators and so have a next method for generating the next item in the iteration. The next method is useful for parallel iteration:

```
>>> data_itr = asia['Dyspnea'].parent_insts_data()
>>> for row in asia['Dyspnea'].parent_insts():
...  print row, ' | ', data_itr.next()
...
('absent', 'false')  |  [0.90000000000000002, 0.10000000000000001]
('absent', 'true')  |  [0.29999999999999999, 0.69999999999999996]
('present', 'false')  |  [0.20000000000000001, 0.80000000000000004]
('present', 'true')  |  [0.10000000000000001, 0.90000000000000002]
```

Chapter 6

Models

6.1 Altering models

The two central ways of altering an existing model are by adding and removing factors. When removing a factor, the factor is specified by its variables and the `remove` method is used.

```
>>> from gPy.Examples import asia
>>> asia.remove(['Bronchitis','Smoking'])
>>> print asia

Bronchitis | TbOrCa |      Dyspnea
           |        |   absent  present
---------- | ------ | -----------------
absent     | false  |     0.90     0.10
absent     | true   |     0.30     0.70
present    | false  |     0.20     0.80
present    | true   |     0.10     0.90


Smoking   |       Cancer
          |   absent  present
--------- | -----------------
nonsmoker |     0.99     0.01
smoker    |     0.90     0.10


Cancer  | Tuberculosis |     TbOrCa
        |              |   false  true
------- | ------------ | -------------
absent  | absent       |    1.00  0.00
absent  | present      |    0.00  1.00
```

```
present | absent      |    0.00  1.00
present | present     |    0.00  1.00


      Smoking
  nonsmoker  smoker
-------------------
       0.50    0.50


TbOrCa |        XRay
       |   abnormal  normal
------ | ------------------
false  |       0.05    0.95
true   |       0.98    0.02


VisitAsia |    Tuberculosis
          |   absent  present
--------- | -----------------
no_visit  |     0.99     0.01
visit     |     0.95     0.05


    VisitAsia
  no_visit  visit
-----------------
      0.99   0.01
>>> type(asia)
<class 'gPy.Models.HM'>
```

In the example immediately above the factor with `Bronchitis` and `Smoking` as variables has been removed. As a result the `asia` object is no longer a `BNM` Bayesian network object, and has become a mere `HM` general hierarchical model object. Nonetheless, since its factors all remain CPTs they are printed out as such.

6.2 Copying models

Some methods alter the model object upon which they are called. Methods which perform conditioning (see Section 6.3) are an important example. If the original model object is still required such methods need to be called on a *copy* of the model, not the model itself.

```
>>> from gPy.Examples import asia
>>> asia_cp = asia.copy()
>>> asia_cp.remove('Cancer')
>>> print asia['Cancer']

Smoking   |       Cancer
          |   absent  present
--------- | -----------------
nonsmoker |     0.99     0.01
smoker    |     0.90     0.10

>>> print asia_cp['Cancer']
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Models.py", line 82, in __getitem__
    return self._factors[frozenset(hyperedge)]
KeyError: frozenset(['a', 'C', 'e', 'c', 'n', 'r'])
```

For all model classes, `model.copy()` returns a copy of the model. However, the default behaviour of `copy` is to make a 'shallow' copy: where the model and its copy share the same *domain*—the same dictionary mapping variables to their set of possible values. In this case, altering the domain of the copy also alters that of the original model. *Conditioning* a model does exactly this; observing a variable to have a particular value removes from the domain all other values associated with that variable.

It follows that if the copy is to be conditioned, or its domain altered in any other way, then the copy needs to be a deep copy. A deep copy is returned by `model.copy(copy_domain=True)`. Altering the domain of a copy created in this way, by conditioning for example, leaves the original unchanged in any way.

6.3 Conditioning models

To alter a model by conditioning on some observation use the `condition` method. This method takes one argument which is a dictionary mapping variables to their observed values. Here's an example of conditioning the `asia` Bayesian network:

```
>>> print asia

Smoking   |     Bronchitis
          |   absent  present
--------- | -----------------
nonsmoker |     0.70     0.30
smoker    |     0.40     0.60


Smoking   |       Cancer
          |   absent  present
--------- | -----------------
```

```
nonsmoker |     0.99     0.01
smoker    |     0.90     0.10


Bronchitis | TbOrCa |      Dyspnea
           |        |   absent  present
---------- | ------ | -----------------
absent     | false  |     0.90     0.10
absent     | true   |     0.30     0.70
present    | false  |     0.20     0.80
present    | true   |     0.10     0.90


      Smoking
  nonsmoker  smoker
-------------------
       0.50    0.50


Cancer  | Tuberculosis |     TbOrCa
        |              |   false  true
------- | ------------ | -------------
absent  | absent       |    1.00  0.00
absent  | present      |    0.00  1.00
present | absent       |    0.00  1.00
present | present      |    0.00  1.00


VisitAsia |    Tuberculosis
          |   absent  present
--------- | -----------------
no_visit  |     0.99     0.01
visit     |     0.95     0.05


    VisitAsia
  no_visit  visit
-----------------
      0.99   0.01


TbOrCa |        XRay
       |   abnormal  normal
------ | ------------------
false  |       0.05    0.95
true   |       0.98    0.02
```

```
>>> asia.condition({'Dyspnea':['absent']})
>>> print asia

Bronchitis | Dyspnea | TbOrCa |
---------- | ------- | ------ | ----
absent     | absent  | false  |   0.90
absent     | absent  | true   |   0.30
present    | absent  | false  |   0.20
present    | absent  | true   |   0.10


Bronchitis | Smoking   |
---------- | --------- | ----
absent     | nonsmoker |   0.70
absent     | smoker    |   0.40
present    | nonsmoker |   0.30
present    | smoker    |   0.60


Cancer  | Smoking   |
------- | --------- | ----
absent  | nonsmoker |   0.99
absent  | smoker    |   0.90
present | nonsmoker |   0.01
present | smoker    |   0.10


Cancer  | TbOrCa | Tuberculosis |
------- | ------ | ------------ | ----
absent  | false  | absent       |   1.00
absent  | false  | present      |   0.00
absent  | true   | absent       |   0.00
absent  | true   | present      |   1.00
present | false  | absent       |   0.00
present | false  | present      |   0.00
present | true   | absent       |   1.00
present | true   | present      |   1.00


Smoking   |
--------- | ----
nonsmoker |   0.50
smoker    |   0.50
```

```
TbOrCa | XRay     |
------ | -------- | ----
false  | abnormal |   0.05
false  | normal   |   0.95
true   | abnormal |   0.98
true   | normal   |   0.02


Tuberculosis | VisitAsia |
------------ | --------- | ----
absent       | no_visit  |   0.99
absent       | visit     |   0.95
present      | no_visit  |   0.01
present      | visit     |   0.05


VisitAsia |
--------- | ----
no_visit  |   0.99
visit     |   0.01
```

Note that the conditioned `asia` is no longer a Bayesian network. This is correct since one of its factors (the one including `Dyspnea`) is no longer a CPT. By default, `gPy` plays safe and makes all conditioned models members of the general `HM` class. If you want to force the class of an object to stay the same even after conditioning set the `keep_class` flag: `model.condition(some_dict,keep_class=True)`.

In general, it is possible to condition by specifying a set of values which are still possible values for a variable (implicitly stating that other values are ruled out). For this reason the values in the dictionary specifying the condition must be a *iterable* of values (e.g. a set, list or tuple). It is easy to get tripped up by this by specifying a string as a dictionary value:

```
>>> asia.condition({'Dyspnea':'absent'})
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Models.py", line 179, in condition
    raise ValueError(
ValueError: Dyspnea has values ('absent', 'present'). 'a' is not one of them
```

6.4 Join forest models

6.4.1 Creating join forest models

To run the probability propagation algorithm ('calibration') on join forests, a join forest is required. If your original is not a join forest model—for example it may be a Bayesian network—then you need to create a join forest from it. Typically, the hypergraph associated with your model will not be decomposable, so this hypergraph will need to be altered before a join forest can be built for it. (Recall that a join forest has the hyperedges of a decomposable hypergraph for its vertices.) In the case of Bayesian networks, unless all variables are mutually independent (hardly typical!), the associated hypergraph will not even be reduced, let alone decomposable.

Join forest models are object of class `gPy.Models.JFM`. Here is an example of `JFM` creation (the output has been altered to break long lines).

```
>>> from gPy.Examples import asia
>>> from gPy.Models import JFM
>>> asia_jfm = JFM(asia,modify=True)
>>> print asia_jfm
Cliques:

Bronchitis | Dyspnea | TbOrCa |
---------- | ------- | ------ | ----
absent     | absent  | false  |   0.90
absent     | absent  | true   |   0.30
absent     | present | false  |   0.10
absent     | present | true   |   0.70
present    | absent  | false  |   0.20
present    | absent  | true   |   0.10
present    | present | false  |   0.80
present    | present | true   |   0.90


Bronchitis | Smoking   | TbOrCa |
---------- | --------- | ------ | ----
absent     | nonsmoker | false  |   0.35
absent     | nonsmoker | true   |   0.35
absent     | smoker    | false  |   0.20
absent     | smoker    | true   |   0.20
present    | nonsmoker | false  |   0.15
present    | nonsmoker | true   |   0.15
present    | smoker    | false  |   0.30
present    | smoker    | true   |   0.30


Cancer  | Smoking   | TbOrCa |
------- | --------- | ------ | ----
absent  | nonsmoker | false  |   0.99
absent  | nonsmoker | true   |   0.99
absent  | smoker    | false  |   0.90
absent  | smoker    | true   |   0.90
```

```
present | nonsmoker | false  |   0.01
present | nonsmoker | true   |   0.01
present | smoker    | false  |   0.10
present | smoker    | true   |   0.10


Cancer  | TbOrCa | Tuberculosis |
------- | ------ | ------------ | ----
absent  | false  | absent       |   1.00
absent  | false  | present      |   0.00
absent  | true   | absent       |   0.00
absent  | true   | present      |   1.00
present | false  | absent       |   0.00
present | false  | present      |   0.00
present | true   | absent       |   1.00
present | true   | present      |   1.00


TbOrCa | XRay     |
------ | -------- | ----
false  | abnormal |   0.05
false  | normal   |   0.95
true   | abnormal |   0.98
true   | normal   |   0.02


Tuberculosis | VisitAsia |
------------ | --------- | ----
absent       | no_visit  |   0.98
absent       | visit     |   0.01
present      | no_visit  |   0.01
present      | visit     |   0.00

Separators:

Bronchitis | TbOrCa |
---------- | ------ | ----
absent     | false  |   1.00
absent     | true   |   1.00
present    | false  |   1.00
present    | true   |   1.00


Cancer  | TbOrCa |
------- | ------ | ----
absent  | false  |   1.00
absent  | true   |   1.00
present | false  |   1.00
present | true   |   1.00


Smoking   | TbOrCa |
--------- | ------ | ----
nonsmoker | false  |   1.00
nonsmoker | true   |   1.00
smoker    | false  |   1.00
smoker    | true   |   1.00


TbOrCa |
------ | ----
false  |   1.00
true   |   1.00


Tuberculosis |
------------ | ----
absent       |   1.00
present      |   1.00

Join Forest:
{ {Bronchitis, Smoking, TbOrCa}, {TbOrCa, XRay}, {Cancer, Smoking,
TbOrCa},
{Tuberculosis, VisitAsia}, {Bronchitis, Dyspnea, TbOrCa}, {Cancer, TbOrCa, Tuberculosis} }
Vertices:
[{Bronchitis, Smoking, TbOrCa}, {TbOrCa, XRay}, {Cancer, Smoking,
TbOrCa},
{Tuberculosis, VisitAsia}, {Bronchitis, Dyspnea, TbOrCa}, {Cancer, TbOrCa, Tuberculosis}]
Lines:
{Cancer, Smoking, TbOrCa} - {Bronchitis, Smoking, TbOrCa}
{TbOrCa, XRay} - {Bronchitis, Smoking, TbOrCa}
{Bronchitis, Dyspnea, TbOrCa} - {Bronchitis, Smoking, TbOrCa}
{Cancer, TbOrCa, Tuberculosis} - {Cancer, Smoking, TbOrCa}
{Cancer, TbOrCa, Tuberculosis} - {Tuberculosis, VisitAsia}
```

If the `modify` flag is not set, and the input model is not decomposable then various exceptions will be raised. Here are two examples of this behaviour:

```
>>> from gPy.Models import RM, JFM
>>> from gPy.Examples import asia
>>> asia_jfm_ex = JFM(asia)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Models.py", line 630, in __init__
    join_forest = JoinForest(hm._hypergraph,modify,True,elimination_order)
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 1321, in __init__
    hypergraph._uforest = hypergraph.join_forest()
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 409, in join_forest
    alpha, beta, gamma, inv_gamma, r = self.maximum_cardinality_search()
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 679, in maximum_cardinality_sea
    raise RedundancyError("%s is not reduced" % self)
gPy.Structures.RedundancyError: { {TbOrCa, XRay}, {VisitAsia}, {Tuberculosis, VisitAsia},
{Bronchitis, Smoking}, {Smoking}, {Bronchitis, Dyspnea, TbOrCa}, {Cancer, Smoking},
{Cancer, TbOrCa, Tuberculosis} } is not reduced
>>> asia_jfm_ex = JFM(RM(asia.copy(),modify=True))
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/home/jc/godot/research/gPy/gPy/Models.py", line 630, in __init__
    join_forest = JoinForest(hm._hypergraph,modify,True,elimination_order)
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 1321, in __init__
    hypergraph._uforest = hypergraph.join_forest()
  File "/home/jc/godot/research/gPy/gPy/Structures.py", line 415, in join_forest
    raise DecomposabilityError("%s is not decomposable" % self)
gPy.Structures.DecomposabilityError: { {TbOrCa, XRay}, {Bronchitis, Smoking},
{Tuberculosis, VisitAsia}, {Bronchitis, Dyspnea, TbOrCa}, {Cancer, Smoking},
{Cancer, TbOrCa, Tuberculosis} } is not decomposable
```

6.4.2 Calibrating join forest models

To calibrate a join forest model just use the `calibrate` method. Note that currently `gPy` does not keep track of whether a `JFM` is calibrated or not. In the following example, the default precision (two) for printing numbers in factors makes some numbers appear to be zeroes when they are not. This can be fixed by setting `gPy.Parameters.precision` to a larger value

```
>>> asia_jfm = JFM(asia.copy(),modify=True)
>>> asia_jfm.calibrate()
>>> print asia_jfm
Cliques:

Bronchitis | Dyspnea | TbOrCa |
---------- | ------- | ------ | ----
absent     | absent  | false  |   0.47
absent     | absent  | true   |   0.01
absent     | present | false  |   0.05
absent     | present | true   |   0.02
present    | absent  | false  |   0.08
present    | absent  | true   |   0.00
present    | present | false  |   0.33
present    | present | true   |   0.03


Bronchitis | Smoking   | TbOrCa |
---------- | --------- | ------ | ----
absent     | nonsmoker | false  |   0.34
absent     | nonsmoker | true   |   0.01
absent     | smoker    | false  |   0.18
absent     | smoker    | true   |   0.02
present    | nonsmoker | false  |   0.15
present    | nonsmoker | true   |   0.00
present    | smoker    | false  |   0.27
present    | smoker    | true   |   0.03


Cancer  | Smoking   | TbOrCa |
------- | --------- | ------ | ----
absent  | nonsmoker | false  |   0.49
absent  | nonsmoker | true   |   0.01
absent  | smoker    | false  |   0.45
absent  | smoker    | true   |   0.00
present | nonsmoker | false  |   0.00
present | nonsmoker | true   |   0.00
present | smoker    | false  |   0.00
present | smoker    | true   |   0.05


Cancer  | TbOrCa | Tuberculosis |
------- | ------ | ------------ | ----
absent  | false  | absent       |   0.94
absent  | false  | present      |   0.00
absent  | true   | absent       |   0.00
absent  | true   | present      |   0.01
present | false  | absent       |   0.00
present | false  | present      |   0.00
present | true   | absent       |   0.05
present | true   | present      |   0.00


TbOrCa | XRay     |
------ | -------- | ----
false  | abnormal |   0.05
false  | normal   |   0.89
true   | abnormal |   0.06
true   | normal   |   0.00
```

```
Tuberculosis | VisitAsia |
------------ | --------- | ----
absent       | no_visit  |   0.98
absent       | visit     |   0.01
present      | no_visit  |   0.01
present      | visit     |   0.00

Separators:

Bronchitis | TbOrCa |
---------- | ------ | ----
absent     | false  |   0.52
absent     | true   |   0.03
present    | false  |   0.41
present    | true   |   0.04


Cancer  | TbOrCa |
------- | ------ | ----
absent  | false  |   0.94
absent  | true   |   0.01
present | false  |   0.00
present | true   |   0.05


Smoking   | TbOrCa |
--------- | ------ | ----
nonsmoker | false  |   0.49
nonsmoker | true   |   0.01
smoker    | false  |   0.45
smoker    | true   |   0.05


TbOrCa |
------ | ----
false  |   0.94
true   |   0.06


Tuberculosis |
------------ | ----
absent       |   0.99
present      |   0.01

Join Forest:
```

```
{ {Bronchitis, Smoking, TbOrCa}, {TbOrCa, XRay}, {Cancer, Smoking, TbOrCa}, {Tuberculosis, V
Vertices:
[{Bronchitis, Smoking, TbOrCa}, {TbOrCa, XRay}, {Cancer, Smoking, TbOrCa}, {Tuberculosis, Vi
Lines:
{Cancer, Smoking, TbOrCa} - {Bronchitis, Smoking, TbOrCa}
{TbOrCa, XRay} - {Bronchitis, Smoking, TbOrCa}
{Bronchitis, Dyspnea, TbOrCa} - {Bronchitis, Smoking, TbOrCa}
{Cancer, TbOrCa, Tuberculosis} - {Cancer, Smoking, TbOrCa}
{Cancer, TbOrCa, Tuberculosis} - {Tuberculosis, VisitAsia}
```

6.4.3 Extracting single variable marginals from join forest models

As a convenience, `gPy` provides the `var_marginal` method which can be used to extract the marginal distribution over a single variable from a `JFM`. `var_marginal` returns a parentless `CPT`. The *assumption* is that the `JFM` in question has already been calibrated.

```
>>> print asia_jfm.var_marginal('Dyspnea')

     Dyspnea
  absent  present
-----------------
    0.56     0.44

>>> print asia_jfm.var_marginal('Bronchitis')

    Bronchitis
  absent  present
-----------------
    0.55     0.45

>>> print asia_jfm.var_marginal('Bronchitis')[['absent']]
0.55
```

Chapter 7

Samplers

7.1 Sampling from unstructured discrete distributions

To sample from arbitrary discrete distributions, `gPy` provides the `gPy.Samplers.MultinomialSampler` class. To construct an object of this class a dictionary mapping each value of the distribution to its probability must be provided:

```
>>> from gPy.Samplers import MultinomialSampler
>>> bias_coin = MultinomialSampler({'h':0.7,'t':0.3})
```

The `MultinomialSampler` class provides a single `sample` method (at time of writing):

```
>>> for i in xrange(30):
...  print bias_coin.sample(),
...
h t h h h t t h h t h h t h h h t h t h h h t h h h t t t h
```

Bibliography

[1] Claude Berge. *Graphs and hypergraphs.* North-Holland, Amsterdam, 1973.

[2] Steffen L. Lauritzen. *Graphical Models.* Oxford University Press, Oxford, 1996.

[3] Robert E. Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing*, 13(3):566–579, August 1984.