Constraints for candidate parent sets

The `score` can accept a simple text file which specifies constraints on the allowed candidate parent sets for each variable. The constraints are specified in disjunctive normal form (an "OR" of "AND"s); so, for each variable, a valid candidate parent set must satisfy only one of the specified constraints. (Note that if conjuctive normal form were allowed, then checking each possible candidate parent set entails solving SAT.)

The constraints are specified in the following format.

- Blank lines are allowed.
- The hash ("`#`") begins comments. Everything on the rest of the line is ignored.
- Each non-blank line contains constraints for one of the variables and is specified as follows.

  ```
  line := <variable_name> [<constraint>]+
  <variable_name> := string
  <constraint> := ( <par>+ )
  <par> := [!]<parent>
  <parent> := string
  ```

In the specification, `<variable_name>` and `<parent>` are both strings which match one of the variable names in the input csv file. An exclamation mark (`!`) before a parent name indicates it is forbidden for that constraint; otherwise, the parent is required. No space should occur between the exclamation mark and the parent name. A space *must* occur after each parent name.

Example

The following example shows constraints for the iris dataset distributed with the source code.

```
# variable names: SepalLength,SepalWidth,PetalLength,PetalWidth,Class
Class ( !PetalWidth PetalLength ) # constraints for Class variable
PetalWidth ( !SepalWidth )
SepalLength ( !PetalLength !Class ) ( SepalWidth )
```

The interpretation is as follows. (Again, recall that a candidate parent set must satisfy *only one* of the constraints.)

- No candidate parent set for `Class` can include `PetalWidth`, and they must all include `PetalLength`.
- No candidate parent set for `PetalWidth` can include `SepalWidth`.
- The candidate parent sets for `SepalLength` must either:

  - Include neither `PetalLength` nor `Class` *or*
  - Include `SepalWidth`

For example, the candidate parent set `{Class, SepalWidth}` is valid for `SepalLength` because it satisfies the second constraint (even though it violates the first one); however, the candidate parent set `{Class, PetalWidth}` *is not* valid for `SepalLength` because it satisfies neither of the constraints.

de Campos-style pruning

In principle, there is no problem combining these constraints with pruning during local score calculations. In essence, subsets which do not satisfy any of the constraints cannot be use to prune supersets.

This has not been thoroughly tested, though, so if problems arise, please create an issue on GitHub.