**Checking Data Integrity with Datalog (DLV)**

**General Instructions**

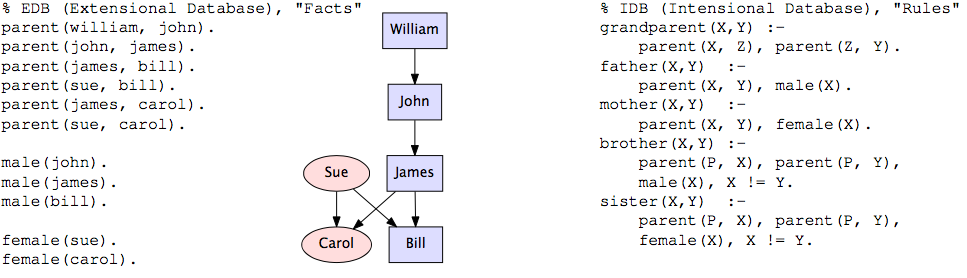
1. To get started, install the DLV system: http://www.dlvsystem.com/dlv on your computer
2. Read the DLV tutorial. There are some notions that will not be easy to understand at first. In particular try and understand the following most important subparts:

**– The Family Tree Example: Predicates, Variables, and Recursion**

**– DLV as a Deductive Database System: Comparison Operators**

1. In the extracted zip file, you will find 2 dlv files, family.dlv and publication.dlv
2. You must answer the following problems in the dlv files by using facts that has been given and make sure you answer them using correct rule name
3. You are not able to add more facts rather than the one that has been given. However, you can add additional rule to support your answer
4. You can test your answer using python script dlvExecutor.py, execute it using python2.7. For example: python2.7 dlvExecutor.py
5. After your answers are ready (family.dlv and publication.dlv) you can submit them all at once using submit.py, execute the script using python 2.7 as well
6. After submission you can check the grading result in coursera web submission page.

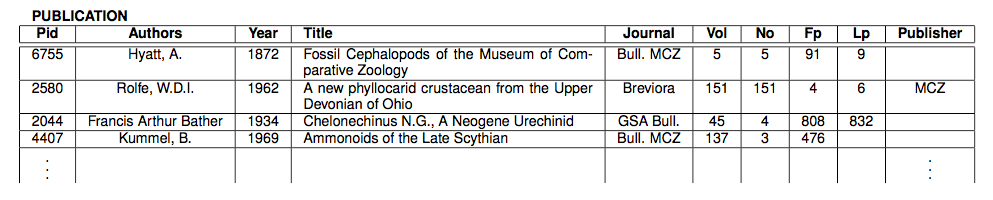
**Part 1: Family (family.dlv)**

****

1. descendant: descendant(X,Y) holds if X is a descendant of Y
2. sibling: sibling(X ,Y ) holds if X and Y are siblings. Hint: X and Y share a parent P
3. write the following integrity constraints (ICs) as “soft constraints” in *denial form*, i.e., write rules that yield  variable bindings that serve as “constraint violation witnesses”:
4. icv\_person\_has\_parent: *Every person must have a parent*. Hint: Look at the IC-rules in the course handouts
5. icv\_person\_has\_father\_mother: *Every person has a father and a mother.* Hint: Same idea as in the Warm-Up. First, write a rule that yields the people for which the IC is satisfied. Then write another rule that reports as an IC-violation those people who are *not* in the answer to that first query.

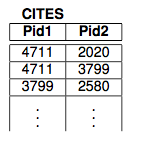
**Part 2: Publication (publication.dlv)**

Consider a “dirty” dataset store publication data in this following relation. The IC-checking capabilities of a database provide a powerful way to detect inconsistencies.



This relational dataset has been transformed into datalog format which you can find in publication.dlv

1. Define the following ICs in *denial form* in Datalog syntax. You can assume that the table is available as a Datalog predicate of the form publication(I, A, Y, T, J, V, N, F, L, P ). Recall that in Datalog, arbitrary (capitalized) names can be chosen as variables, since it is the argument *position* that determines which attribute/column is meant.
2. icv\_pid\_key: The publication identifier Pid is a *key*, i.e., if a row agrees with another row on the key attribute Pid, then it also agrees on all other attributes (i.e., the “two” rows are in fact one and the same). As usual, your rule should return the IC-violations.
3. icv\_journal\_publisher: Every Journal has a single Publisher. Like (FD-1), this is a *functional dependency*. It is sometimes written as Journal → Publisher.
4. ncv\_firstpage\_lastpage: The last page Lp cannot be smaller than the first page Fp. Note: This numerical constraint can be evaluated independently on each row.
5. Now consider that an additional table cites(P1, P2) is given which records pairs of publication P1, P2, where P1 is citing P2. Define the following IC in denial form:



1. icv\_cited\_publication: Every cited publication in CITES also occurs in PUBLICATION. This is an *inclusion dependency* and is usually written in the form: CITES[Pid2] ⊆ PUBLICATION[Pid].
2. ncv\_p1\_greater\_p2: If P1 cites P2, then P2’s year of publication cannot be greater than P1’s year of publication.