Exercise 1 (Procedural Semantics): (5 (only V3M) points)

Consider the following constraint logic program $\mathcal{P}$ for the computation of the Fibonacci numbers together with the constraint theory $CT_{FD}$ from Example 6.1.4 of the lecture (i.e., $CT_{FD}$ consists of all true formulas on integers):

$$
\text{fib}(0,0).
\text{fib}(1,1).
\text{fib}(N,F) :- N > 1, N_1 = N - 1, N_2 = N - 2, \text{fib}(N_1,F_1), \text{fib}(N_2,F_2), F = F_1 + F_2. 
$$

Please write down both successful computations of the query $G = \{\neg\text{fib}(X,1)\}$ w.r.t. $\mathcal{P}$ and $CT_{FD}$, i.e., a sequence of configurations of the form $(\text{fib}(X,1), \text{true}) \vdash \mathcal{P} \ldots \vdash \mathcal{P} (\square, CO)$. As in Example 6.1.15 from the lecture, you may leave out the negations in the queries and simplify your constraints after every computation step. Moreover, please give the computed answer $P[\mathcal{P}, CT_{FD}, G]$.

Solution:

First solution:

$$
\vdash \mathcal{P} (\square, \text{true} \land \text{fib}(X,1) = \text{fib}(1,1)) \quad X = 1
$$

Second solution:

$$
\vdash \mathcal{P} (\sqcup, \text{true} \land \text{fib}(X,1) = \text{fib}(1,1)) \\
\vdash \mathcal{P} (N > 1, N_1 = N - 1, N_2 = N - 2, \text{fib}(N_1,F_1), \text{fib}(N_2,F_2), F = F_1 + F_2, \text{true} \land \text{fib}(X,1) = \text{fib}(N,F)) \\
\vdash \mathcal{P} (N_1 = N - 1, N_2 = N - 2, \text{fib}(N_1,F_1), \text{fib}(N_2,F_2), F = F_1 + F_2, N > 1 \land X = N \land 1 = F) \\
\vdash \mathcal{P} (N_2 = N - 2, \text{fib}(N_1,F_1), \text{fib}(N_2,F_2), F = F_1 + F_2, N_1 = N - 1 \land N > 1 \land X = N \land 1 = F) \\
\vdash \mathcal{P} (\text{fib}(N_1,F_1), \text{fib}(N_2,F_2), F = F_1 + F_2, N_2 = N - 2 \land N_1 = N - 1 \land N > 1 \land X = N \land 1 = F) \\
\vdash \mathcal{P} (\text{fib}(N_2,F_2), F = F_1 + F_2, \text{fib}(N_1,F_1) = \text{fib}(1,1) \land N_2 = N - 2 \land N_1 = N - 1 \land N > 1 \land X = N \land 1 = F) \\
\vdash \mathcal{P} (F = F_1 + F_2, \text{fib}(N_2,F_2) = \text{fib}(0,0), N_1 = 1 \land F_1 = 1 \land N_2 = 0 \land X = 2 \land N = 2 \land 1 = F) \\
\vdash \mathcal{P} (\square, F = F_1 + F_2 \land N_2 = 0 \land F_2 = 0 \land N_1 = 1 \land F_1 = 1 \land X = 2 \land N = 2 \land 1 = F)
$$

$P[\mathcal{P}, CT, G] = \{\text{fib}(2,1), \text{fib}(1,1)\}$
Exercise 2 (SLD Trees With CLP): (5 (only V3M) points)

Consider the following Prolog program for the well-known Ackermann function with constraints from $CT_{FD}$:

```prolog
:- use_module(library(clpfd)).

ack(0, M, R) :- !, R #= M+1.
ack(NP, 0, R) :- !, NP #= NP-1, ack(NP,1,R).
ack(NP,MP,R) :- NP #= NP-1, MP #= MP-1, ack(NP,M,RP), ack(N,RP,R).
```

Please give a graphical representation of the full SLD tree for the query `- ack(1, 1, Z).` You can restrict your representation to those parts of the tree where the conjunction of constraints is satisfiable (i.e., do not show the parts where the edges contain unsatisfiable constraints). Please also give the answer substitution.

Solution: 

a) 

2
ack(1, 1, Z)
NP=1∧MP=1∧Z=R

N=NP-1, M=MP-1, ack(NP,M,RP), ack(N,RP,R)
N=0∧NP=1∧MP=1∧Z=R

M=MP-1, ack(NP,M,RP), ack(N,RP,R)
M=0∧N=0∧NP=1∧MP=1∧Z=R

ack(NP,M,RP), ack(N,RP,R)
R1=RP∧NP1=1∧M=0∧N=0∧NP=1∧MP=1∧Z=R

!,N1=NP1-1, ack(N1,1,R1),ack(N,RP,R)
R1=RP∧NP1=1∧M=0∧N=0∧NP=1∧MP=1∧Z=R

N1=NP1-1, ack(N1,1,R1),ack(N,RP,R)
N1=0,R1=RP∧NP1=1∧M=0∧N=0∧NP=1∧MP=1∧Z=R

ack(N1,1,R1),ack(N,RP,R)
M2=1∧R2=R1∧N1=0∧R1=RP∧NP1=1∧M=0∧N=0∧NP=1∧MP=1∧Z=R

!,R2=M2+1,ack(N,RP,R)
M2=1∧R2=R1∧N1=0∧R1=RP∧NP1=1∧M=0∧N=0∧NP=1∧MP=1∧Z=R

R2=M2+1,ack(N,RP,R)
R2=2∧M2=1∧R1=2∧N1=0∧RP=2∧NP1=1∧M=0∧N=0∧NP=1∧MP=1∧Z=R

ack(N,RP,R)
M3=2∧R3=R∧R2=2∧M2=1∧R1=2∧N1=0∧RP=2∧...∧Z=R

!,R3=M3+1
M3=2∧R3=R∧R2=2∧M2=1∧R1=2∧N1=0∧RP=2∧...∧Z=R

R3=M3+1
M3=2∧R3=3∧R=3∧R2=2∧M2=1∧R1=2∧N1=0∧RP=2∧...∧Z=3
□

The answer substitution is \{Z = 3\}.

Exercise 3 (Programming in CLP): (3+4 (only V3M) points)

Important: In addition to handing in the solution on paper, please also mail your the solutions for this exercise to lp15-hiwis@i2.informatik.rwth-aachen.de. Indicate your immatriculation numbers in the subject of the mail and inside the Prolog file.
a) Write a Prolog query to find all triples \((a, b, c) \in \mathbb{N}^3\) such that \(a^2 + b^2 = c^2\) and \(a + b + c = 1000\). Do not define any new predicates. You may assume that the clpfd module was loaded before (e.g., by entering `use_module(library(clpfd)).` in the Prolog shell).

b) The mathematical puzzle Graph-Sudoku is a generalization of the popular riddle Sudoku. We are given a Graph \(G = (V, E)\) with vertices \(V\) and edges \(E\) and we want to find an assignment \(V \mapsto \{1, \ldots, n\}\) such that no two neighboring nodes in the graph have the same number. If one replaces the vertices by these numbers, we call it a valid Graph-Sudoku.

Your task is to implement a CLP predicate `gsudoku/2` in the constraint theory \(CT_{FD}\). Let a graph be given by a list `Edges` of edges `e(X,Y)`, let \(N\) be the maximal number to be used, and let \(V\) be a list of pairwise different variables, representing the vertices. The query `?- gsudoku(Edges,N), label(V).` should be provable iff the variables used in `Edges` can be instantiated to a valid Graph-Sudoku in our representation. In addition, if `gsudoku(Edges,N), label(V).` is provable, then variable entries in the individual edges should be instantiated by the answer substitution.

For example, the list `[e(X,Y), e(Y,Z), e(Z,X)]` represents the following graph:

```
  X
 /\  \
 |  |  \
|  |  |
Y-+----Z
```

The query `?- gsudoku([e(X,Y), e(Y,Z), e(Z,X)], 2), label([X,Y,Z]).` should fail (there is no valid assignment from \([X,Y,Z]\) to \([1,2]\)). However, the query `?- gsudoku([e(X,Y), e(Y,Z), e(Z,X)], 3), label(X,Y,Z).` should have the result \(X = 1, Y = 2, Z = 3\) (or any other permutation).

Solution:

a) % assume this was executed: use_module(library(clpfd)).
   A in 0..1000, B in 0..1000, C in 0..1000, A*B*C #= 1000, A*A+B*B #= C*C, label([A,B,C]).

b) :- use_module(library(clpfd)).
   gsudoku([],_).
   gsudoku([ e(F,T) | XS],N) :- F in 1..N, T in 1..N, F #\= T, gsudoku(XS,N).