Equality logic with uninterpreted functions

Implement a decision procedure for equality logic with uninterpreted functions. The grammar of the above logic is defined as follows:

$$\mathit{formula} : \mathit{formula} \ \wedge \ \mathit{formula} \mid \neg \mathit{formula} \mid (\mathit{formula}) \mid \mathit{atom}$$
$$\mathit{atom} : \mathit{term} \ = \ \mathit{term} \mid \mathit{predicate\text{–}symbol}\ (\mathit{list\ of\ terms})$$
$$\mathit{term} : \mathit{identifier} \mid \mathit{function\text{–}symbol}\ (\mathit{list\ of\ terms})$$

To check the satisfiability of an input formula, first create a DAG. You can use parser generator tools such as Flex+Bison (C/C++) and Antlr (Java) to generate a DAG. After generating the DAG, use the algorithm based on the union-find data structure to compute the congruence class of each node in the DAG. Finally, verify that the inequalities don't belong to the same congruence class for satisfiability.

See Chapter 9.3 from the Calculus of Computation book for more details.

Here are some of the test cases. Each line is a different test case.

1. f(f(f(a))) = a & f(f(f(f(f(a))))) = a & f(a) != a
2. f(a, b) = a & f(f(a,b),b) != a
3. f(x, y) = f(y, x) & f(a,y) != f(y,a)
4. f(g(x)) = g(f(x)) & f(g(f(y))) = x & f(y) = x & g(f(x)) != x
5. f(f(f(a))) = f(f(a)) & f(f(f(f(a)))) = a & f(a) != a
6. f(f(f(a))) = f(a) & f(f(a)) = a & f(a) != a
7. f(f(x1, y1) , f(x2, y2)) != x1 & f(x1, y1) = x1 & f(x2, y2) = y2 & y1 = y2
8. f(a, b) = g(f(c))

Siddharth Nayak and Puneet Kumar have kindly agreed to give a tutorial on Flex+Bison and Antlr tools if needed. Please feel free to talk to them if you need help. You can use any programming language of your choice. Submit your implementation and instructions on how to compile and run your implementation. You can do it in a group of up to three students. The group for this assignment must be different from all your previous groups. All students in a group need to understand the implementation thoroughly. The marks will be given based on the average understanding of the group.

Bonus marks:
You can get up to two bonus marks if you provide good test cases.