# Final Exam

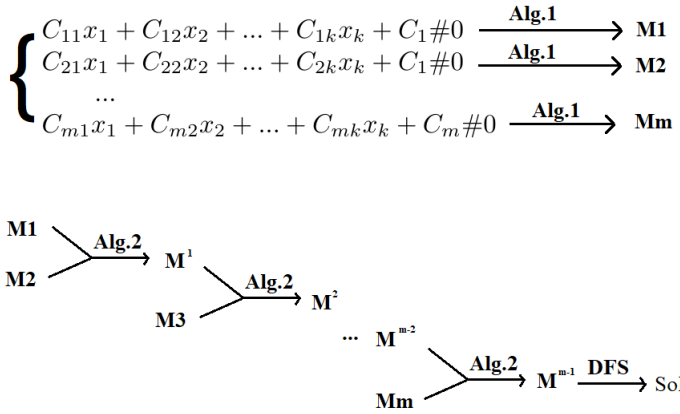
Rusu Wu
11694764

**1.**

```
//Implement a function that returns Cmax

void CacCmax(int c1, int c2, int c3, int c) {
    int Cmax = 0;
    for (int d = 0; d <= 1; d++) {
      for (int d1 = 0; d1 <= 1; d1++) {
        for (int d2 = 0; d2 <= 1; d2++) {
          for (int d3 = 0; d3 <= 1; d3++) {
            int max = abs(c1 * d1 + c2 * d2 + c3 * d3);
            if (max > Cmax) Cmax = max;
}
}
}
}
    return Cmax;
}

//Implement a function that returns the value KC from a given constant C>=0.

int CacKc(int C) {
    int Kc = 0;
    while (C > 0) {
        C = C/2;
        Kc++;
}
    return Kc;
}

//Implement a function that returns the value bi from a given constant C>=0 and i,

int CacBi(int C, int i) {
    if (i == CacKc(C) + 1) return 0;
        for(i=i-1;i>=0;i--){
                C = C/2;
}
    return C % 2;
}
```

**2.** During the implementation, by useing the $Algorithm1$, we firtly generted finite automata $M1$ and $M2$ based on the two constraints. In other words, we converted two equations to two labeled graphs with certain states through $Algorithm1$. Then, by applying $Algorithm2$, we can get cartesian product $M$ from finite automata $M1$ and $M2$. $M$ also can be seen as a labeled graph. Finally, we used DFS to get the solutions of $M$; the solution is exactly the solution of the two constraints.

We can generalize the algorithm to a general LD instance Q as the figure shown below;




Generalization Process]

Firstly, we use the $Algorithm1$ to generate labeled graphs for all of the constraints from 1 to m. Hence, we can gain $m$ labeled graphs from $M1$ to $Mm$.

Secondly, we can use $Algorithm2$ to get cartesian product $M^1$ from $M1$ and $M2$($M^1$ is also a labeled graph).

Thirdly, we continue use $Algorithm2$ to get cartesian product $M^2$ from $M^1$ and $M_3$. $M^2$ is our updated cartesian product now. Next time we do the same process with $Algorithm2$; we get the updated cartesian product from $M^i$ and $Mi+2$ until we get a final cartesian product $M^{m-1}$ from $M^{m-2}$ and $Mm$ by applying the $Algorithm2$, $i$ is the current number of updated cartesian product here.

Finally, we get the final labeled graph $M^{m-1}$ of the general LD instance, and we use $DFS$ on the graph. If we can find a path from the initial to accepting, we can get the solution of the LD instance. Otherwise, the LD instance does not have a solution.

## 3.

## Defining Similarity of Two General Linear Diophantine Instances

### Introduction

Solving a linear Diophantine(LD) constraints is a NP-complete problem, and thousands of sientists have made efforts on exploring the solution method.

linear diophantine constraints can be defined as: for some n,and m;

$$C_{11}x_1 + C_{12}x_2 + ... + C_{1n}x_n + C_1\#0$$
$$...$$
$$C_{m1}x_1 + C_{m2}x_2 + ... + C_{mn}x_n + C_m\#0$$

For every basic formula $C_{k1}x_1 + C_{k2}x_2 + ... + C_{kn}x_n + C_k = 0, (0 \leq k \leq m)$, no matter trough manual or algorithm, we can firstly convert it into automaton graph through saturating the set of transition and states[1]; After all the convertions of basic f, we can do cartesian product to combine all of the graphs into one graph and then gain the solution through runing the graph form start to the accepting.

In this case, since LD constraint can be converted into graph,so the similarity of LD constraints can be defined as the similarity of graphs.

### Measuring the Similarity

When there is a LD that has certain solution, we can make sure that we have a accepting state in the graph.

To begain with, we should explore the the properties of a graph that is converted from a linear diophantine constraints.

1)In the process of runninng a graph from initial to the accepting state, assumed that we will run 5 steps from one node to and node and finally we abtain the accepting state, the bigist solution of one of the unknow value is less that or equal to $2^5 - 1$. In other words,the largest solution value of one of the unknowns such as $(x_1, x_2...)$ are closely related to the total steps from initial to the end.

2)Assuming that the current step is $i$, the current maximum value of one of the unknowns is $2^i - 1$. Also, the range of the largest unknow would be in the range of $2^i - 2^{i-1}$. And after that, whenever the graph move from one edge through one edge, the current maximum value of one of the unknowns may increase to $2^{i+1} - 1$.

In this case,we can enumerate the edges routes that hve the same step number, and defind their simularity because the final largest solution is related to the step number when it gets accepting.

However, since the running route about going through which edge is undertermined, it is hard to obtain the edge routes that the two compared graphs run from initial to the accepting specifically (in a certain step number), expecially for two very large graphs.

Hence, I defined that we can compare the number of similar edges from initial node to the end node,and defied a equation as the measurement of two graphs. The states of nodes do not matter here becaudse they do not affect the solution of LDs. In this case, I need to extract the most common graph of two graphs by comparing them. I would show how to extract a common graph in example part.The extracted graph can is the overlapping part of the two graphs regardless of the node states and the edge labels from the initial node and accepting node. Such that we can modulate $LD_1$'s graph and $LD_2$'s graph from the initial to the end with the same number of steps.

Hence I defined the simulary metric as:

$$Sim. = \frac{2 * E_n(A \cap B)}{E_n(A) + E_n(B)} \tag{1}$$

In the above formula, $A$ and $B$ are the two graphs that are converted from $lD_1$ and $LD_2$, $A \cap B$ means the graph that is extracted from graph $A$ and $B$. $E_n(graph)$ is the edge number of a graph. In this formula, I have not considered the node number since the node number has limited effetiness on the final solution of $LD$ contraints.

Another case is that, sometimes the two graphs of the two $LDs$ have a same route frome initial to the accepting, so as the extracted graph. In this case, we could not say that the two $LDs$ have a same $Similarity$ value. Therefore, we should consider the labels of extracted graphs as one of factors when we count for simularity.

In the extracted graph, named all of the edge as $(1, 2...., E_n(A \cap B))$, $E_n(A \cap B)$ is the total edge number of the extracted graph. The label of $A$ represent in $edge_1$ defined as $l_A(a_{11}, a_{12}, ...a_{1n})$ n is the number of unknown value which is defined at basic formula $C_{11}x_1 + C_{12}x_2 + ... + C_{1n}x_n + C_1 = 0$. Similary, The label of $B$ is represented as $l_B(b_{11}, b_{12}, ...b_{1n})$.

The final solutions of the extracted graph are closely linked to the lables. For example, when we run the extracted graph, in the same $i$ step,the graph would reach accepting state so as $LD_1$ and $LD_2$. During the runing, such as the current runing step is $k-1$, it only has one edge to the next state, and the lable is $(a_{e1}, a_{e2}, ...a_{en})(b_{e1}, b_{e2}, ...b_{en})$, e is the edge name here$(1 \leq e \leq E_(A \cap B)$. Assume the the current values of unknowns are $(x_1, x_2, x_3...x_n), (y_1, y_2, y_3...y_n)$ When the extracted graph run trought the edge, the unknow value would become bollows;

$$(x_1 = x_1 + 2^{ka_{e1}}, x_2 = x_2 + 2^{ka_{e2}}, ...x_n = x_n + 2^{ka_{en}})$$
$$(y_1 = y_1 + 2^{kb_{e1}}, y_2 = y_2 + 2^{kb_{e2}}, ...y_n = y_n + 2^{kb_{en}})$$

The solution of $A$ and $B$ would be affected by $l_A$ and $l_B$ when run through the edge, so I defined a disimilaity parameter[2] of the extracted graph as follow.

$$Disim. = \frac{1}{E_n(A \cap B)} \sum_{e=1}^{E_n(A \cap B)} \sum_{1}^{n} \frac{\max \mid (a_{e1} - b_{e1}) \mid + \max \mid (a_{e2} - b_{e2}) \mid + ... + \max \mid (a_{en} - b_{en}) \mid}{n} \tag{2}$$

Sometimes, on one edge, there can be several labels of $A$ or $B$, so I use the maxmum of $(a_{en} - ben)$ to meature the disimilarity.

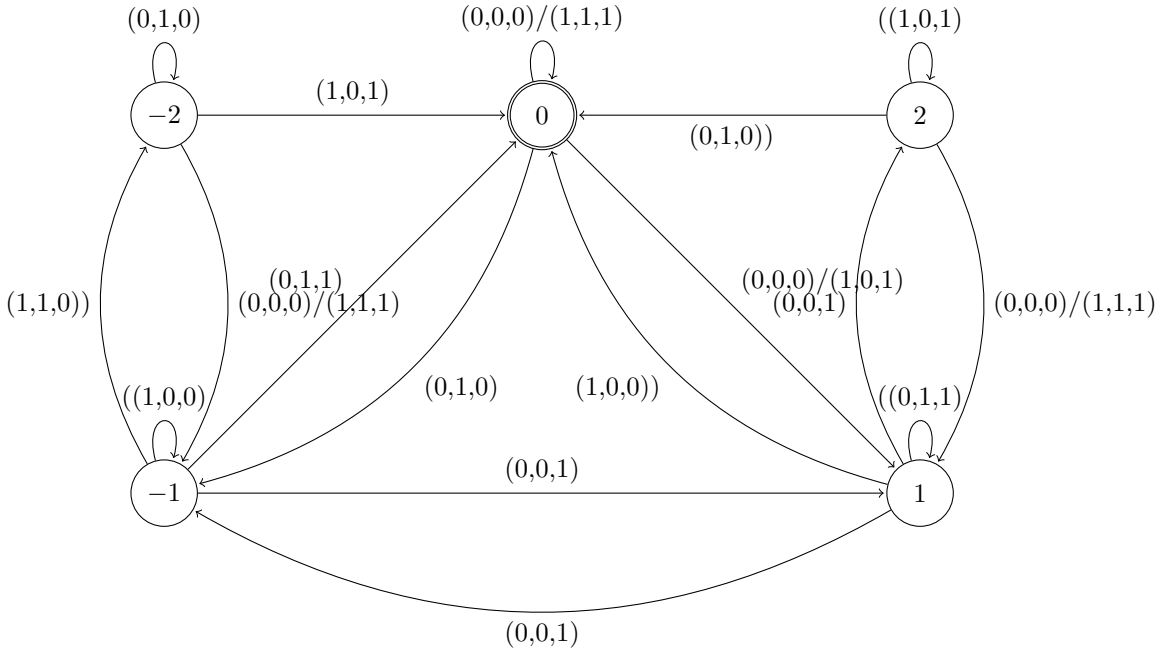
Finally, the $Similarity$ of $LD_1$ and $LD_2$ can be difined as the similarity of Graph $A$ and Graph $B$ as following.

$$Similarity(LD_A, LD_B) = Sim. - Disim. \tag{3}$$

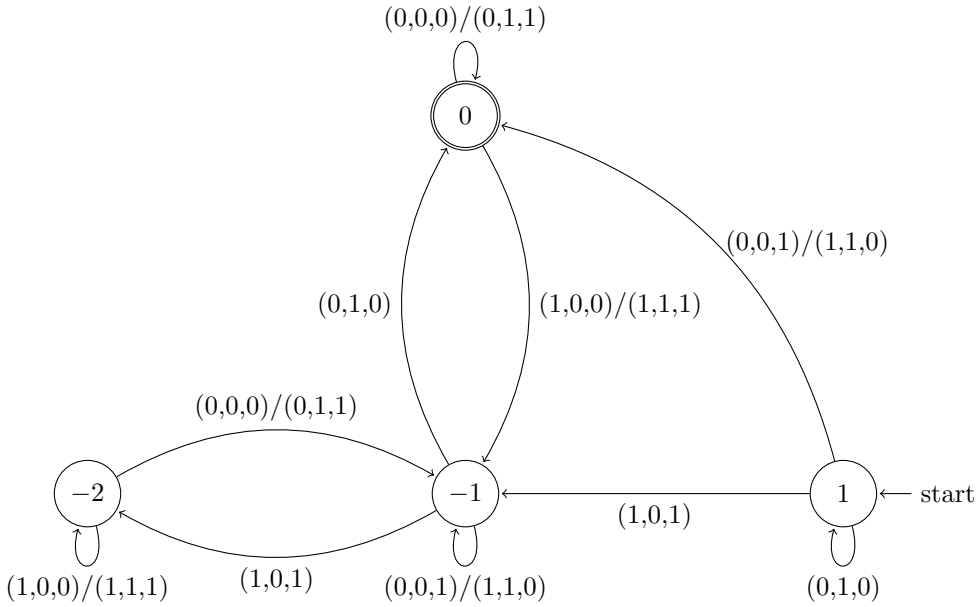
Then Range of $Sinmilarity$ is in the range of $-1 < Similarity \leq 1$. I defined that, if $Similarity \geq 0$,the two graph has different level of similarity base on the value. If $Similarity = 1$, the two graphs are totally same.If $Similarity \leq 0$, then we could say that the two $LD$s has less similarity.

**Example of Measuring Similarity**

Take a 1-constraint LDs as example.
$LD_1$: $-x_1 - 2x_2 + 3x_3 + 1 = 0$
$LD_2$: $2x_1 - x_2 + x_3 + 1 = 0$
Convert them into graphs as following[1]:

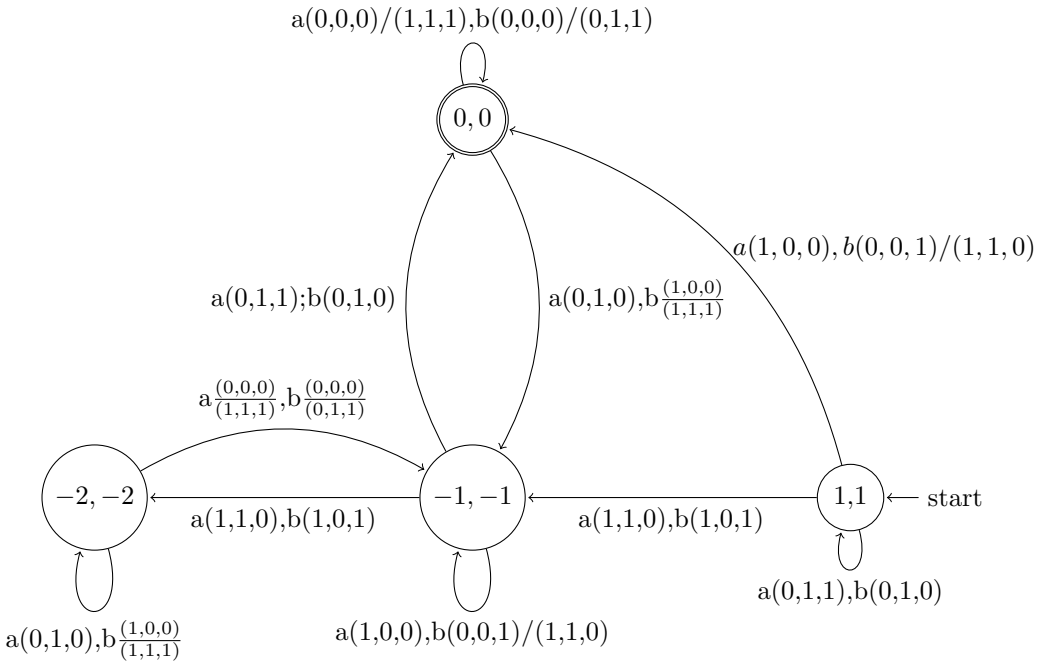



Above is the graph $A$ of $LD_1$: $-x_1 - 2x_2 + 3x_3 + 1 = 0$




Above is the graph $B$ of $LD_2$: $2x_1 - x_2 + x_3 + 1 = 0$

Then, we extracted the largest graph by comparing the step number from intial to the end as follow.




Above is the graph extracted from $A$ and $B$, and then we can see that $E_n(A \cap B) = 10$, $E_n(A) = 17, E_n(B) = 10$. Hence $Sim. = 0.741$

Then After computation,we can gain that $Disim. = 0.633$, so the $Similarty$ is :

$$Similarity(LD_1, LD_2) = Sim. - Disim. = 0.741 - 0.633 = 0.108$$

Then I could say that $LD_1$ adn $LD_1$ has 10.8% similarity.

**Applications of the Similarity Metric**

Since many irragular events in nature can be roughly represented by linear diophantine constraints, reasearching on the similarity of LDs can help us understand how the events are going by comparing them. Currently, big data mining and prediction of event behaviors are very popular research directions, and the similarity metric would be useful in both areas.

For example, the repid development of the internet brings properity to e-commerce; many people would leave their purchase records on website. Since every individual is different, we can use linear diophantine constraints to represent their purchase behaviors. Based on the simirality metric and their purchase characterstic equation, we can divide people into many categories. In this case, we can recommend different products on the Web homepage or adopt different selling methods to different people to increase sales.

Another instance is about predition. The trend of stocks can be affected by many factors such as market demand, future policies, company's profitability and so on. We all know that it is hard to predict the trend of stocks. However, since the sensitivity of stocks to these factors is not the same, we can convert their trands to be linear diophantine constraints. In this case, we can compare the target stock with other stocks by applying the similarity metric. If there is a stock has a high similarity score with the target stock, we can use the old data to predict the fucture trend of the target stock.

## Conclusion

In this essay, based on knowing that $LD$ instances can be convert into graphs, I define a similarity measuring method of two label graphs in order to measure the similarity of two $LD$ instances. The first step is to extract the commom graph considering egdes from initial to the accepting base on the running step. By comparing their commom edge number of the extracted graph and the edge number of the two $LD$s' graphs, we can get a $Similarity$ score as well as a $Disimilarity$ score. After that, we can computer the distance between edge labels of the two $LD$s' graphs in the extracted graph to get a disimilarity score. Then, by considering the $Similarity$ score and $Disimilarity$ score, we can tell the Similarity of the two graphs as well as the two $LD$ instances. At the end, I summarized the applications that this similarity metric can adapt to, including big data mining and prediction of event behaviors.

However, there are several weaknesses within the measuring method. Firstly, the method for extracting common graph is still too fuzzy, and this needs future work to explord it. Furthermore, no matter the process of extracting common graph or the process of computing distance between labels, they are too complex. Future work is to disign algrithms for achieving efficiency.

## Reference

[1]M. Obaid, Automata Theory for Presburger Arithmetic Logic, 18-Jun-2002. [Online]. Available: https://people.mpi-inf.mpg.de/ hillen/documents/6_Presburger.pdf. [Accessed:May-2021].

[2]P.-A. Champin and C. Solnon, "Measuring the Similarity of Labeled Graphs," Case-Based Reasoning Research and Development, pp. 80–95.