



Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

The Frame Problem

A Final Year Project Report

Written by

Aayush Choudhary

17th may 2023

Under the supervision of **Dr. Tim Fernando**, and submitted to the School of Computer Science and Statistics for module CSU44E02 (Computer Engineering Project)

at Trinity College Dublin.

Acknowledgement

I would like to give a big thank to my supervisor, Dr Tim Fernando, who guided and provided support throughout the entire duration of this project. I am deeply indebted to them for their insights and expertise in this area. I would also like to extend my gratitude to Dr. Alessio Benavoli for giving his time for me to demonstrate and show my work on this project. I would also like to Thank my parents for always supporting me specially during my undergraduate career, and to my friends I made along the way for a memorable time.

Abstract

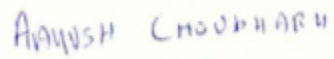
The frame problem is a very concerning obstacle in AI which fails to express changes or non-changes caused by a large number of intuitively non-effects of actions in First-order logic. The frame problem analysis randomness induced in a system that is deterministic in nature from effects of other actions in a system that is non-deterministic in nature.

Deterministic systems can be superposed to a system of actions where all changes or non-changes caused by effects of all actions are merged. A set of system of actions that is deterministic in nature when superposed with another set of system of actions that is either deterministic or non-deterministic in nature becomes a system of actions that is non-deterministic nature. The 13 temporal relations developed by Allen and Ferguson provides an approach to frame problem.

Action languages are formal models for talking about effects of actions, a transition system can be used to formalize actions. In a transition system of superposed action signatures actions are defined on a set of truth-valued function known as Elementary actions. The inertial approach to the Frame Problem assumes there is no other elementary action for non-effects.

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.



Aayush Choudhary

Contents

Introduction	6
1. Allen Finite state and Allen interval relation	7
1.1 Bounding life of two events	7
1.2 Allen Relations	9
1.3 Superposing automata	11
1.4 Python code for Superposition	13
2. Action languages	16
2.1 Action signatures.	16
2.2 Transition Systems	17
2.3 Elementary Actions	18
2.4 Python code for Action languages	19
3. The Frame Problem for elementary actions	21
3.1 Bounding life of two events as an action language transition system.	21
3.2 Python code Relations	23
Security and privacy regarding Frame Problem	24
Conclusion	26
Bibliography	27

Introduction

The Frame Problem: A Python notebook. The aim of this project is to understand the various approaches to Frame problem by working them out over finite transition systems, encoded in Python. The Frame Problem is a classic problem in AI concerning the changes and non-changes that arise from performing actions. The frame problem in simple words frame problem is the problem of representing the effects of action in First-order logic without having to represent explicitly a large number of intuitively obvious non-effects. For example, there is a robot who has to open a door but there are many factors evolve like whether the door is locked or not, how is the handle of door (shape and type), etc. Now if there is change in door locked or not locked status robot still has to know how is the handle of door which is obviously in same shape and type. Motivation behind dealing with this particular problem is that automation will make human work minimal and frame problem is an obstacle in it.

The project I first went along to study Allen Finite state and Allen interval relation. Allen's interval algebra is a calculus for temporal reasoning that was introduced by James F. Allen in 1983. I wrote some python scripts to represent a Allen finite state model as a NFA(non-deterministic finite automata) using automata and visual_automata library in python , I also studied and applied superposition in this section of my project. Then I studied action languages as a part of this project. Action languages are formal models of parts of the natural language that are used for talking about the effects of actions. Published in 1998Linkoping University Electronic Press. I also represented a model of an action signature using automata and visual_automata library in python. Lastly I modeled the Allen finite state model with action languages and represented it using automata and visual_automata library as NFA(non-deterministic finite automata) which resulted in an interesting conclusion.

1 Allen Finite state and Allen interval relation

1.1 Bounding life of two events

We have taken life bound of two events here for a and b as shown in diagram below. Events a and b both are deterministic in nature individually. Here we have an Event with 3 states u, li and d signifying unborn, living and dead respectively and with two actions l and r signifying live and rest respectively.

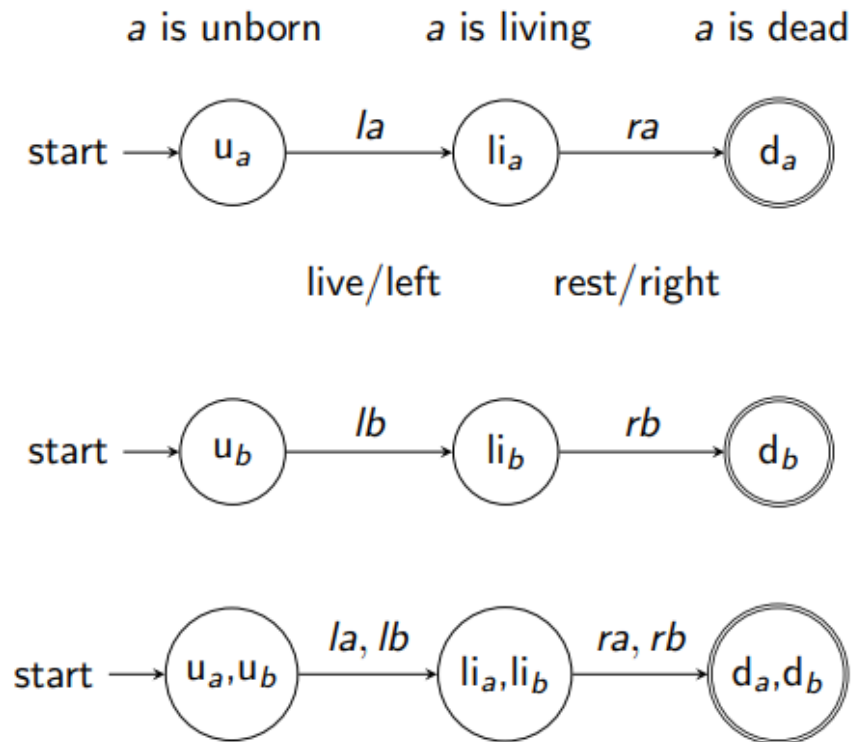


Fig 2.1.1

Both of the event's a and b are happening simultaneously which could happen as shown in Fig 1.1.1 i.e., both a and b are in unborn state initially then both live and move to a living state, at the end both a and b rest and are in dead state but this is only true when both events live and rest at the same time period which is highly unlikely to happen in most of times in real life scenarios. Fig 1.1.2 is a Finite state machine for both events a and b are superposed resulting in all the ways possible in which both a and b can happen.

One of the possible ways is highlighted in Fig 1.1.2. Here initially both a and b are in unborn state, then an action takes places in time period 1 i.e., b is born only (l_b) which makes a in unborn state and b in living state, then another action takes places in time period 2 i.e., b dies only (r_b) which makes a in unborn state and b in dead state, then another action takes places in time period 3 i.e., a is born only (l_a) which makes a in living state and b in dead state, finally last

set of action takes places in time period 4 i.e., a dies only (r_a) which makes a in dead state and b in dead state.

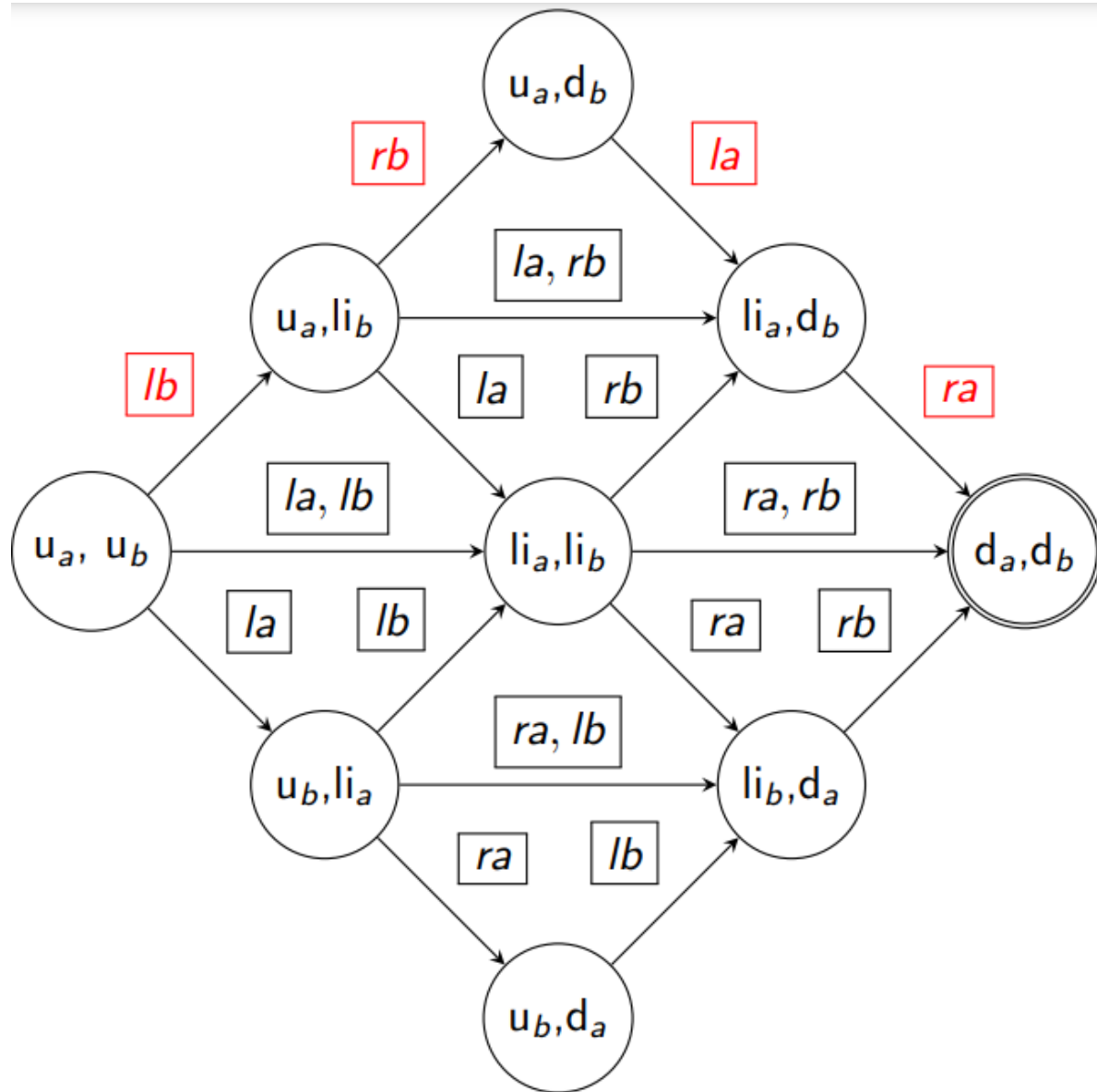


Fig 2.1.2

There can be any possible combination from the figure 1.1.2 like another way is la , then lb and then ra , rb together. Both events a and b are deterministic in nature individually when superposed the finite state machine becomes a non-deterministic system in nature.

1.2 Allen Relations

13 Allen Temporal relations works as representational framework for the frame problem which then combined with various approaches to frame problem gives out powerful results. For just very simple events such as a and b when superposed together to study changes and non-changes caused by effects of all actions are merged into a non-deterministic system. One of the challenges faced are that the system becomes so difficult to analyze. In this case itself we have two very simple events but when superposed becomes this complex Finite state machine. When we consider complex events that takes place in real life which are way complex events compared to the events we have taken as example in Fig 1.1.1 and to then analyze all the changes and non-changes caused by effects of all actions becomes kind of impossible. This is where Fin Allen relations comes into picture. These Allen relations represented on the right-side of Fig. 1.2.1 very efficiently represents the axiomatization of time period and relationships between actions and events and their effects that are represented on the left-hand side of Fig. 1.2.1. When the systems become more and more complex, we can represent various actions under these 13 categories and analyze them easily. As you can see each way possible in our Finite state Machine is represented by a relation in Fig 1.2.1. One of the ways is highlighted i.e., la, ra, lb, rb each at different time periods is represented by a before b.

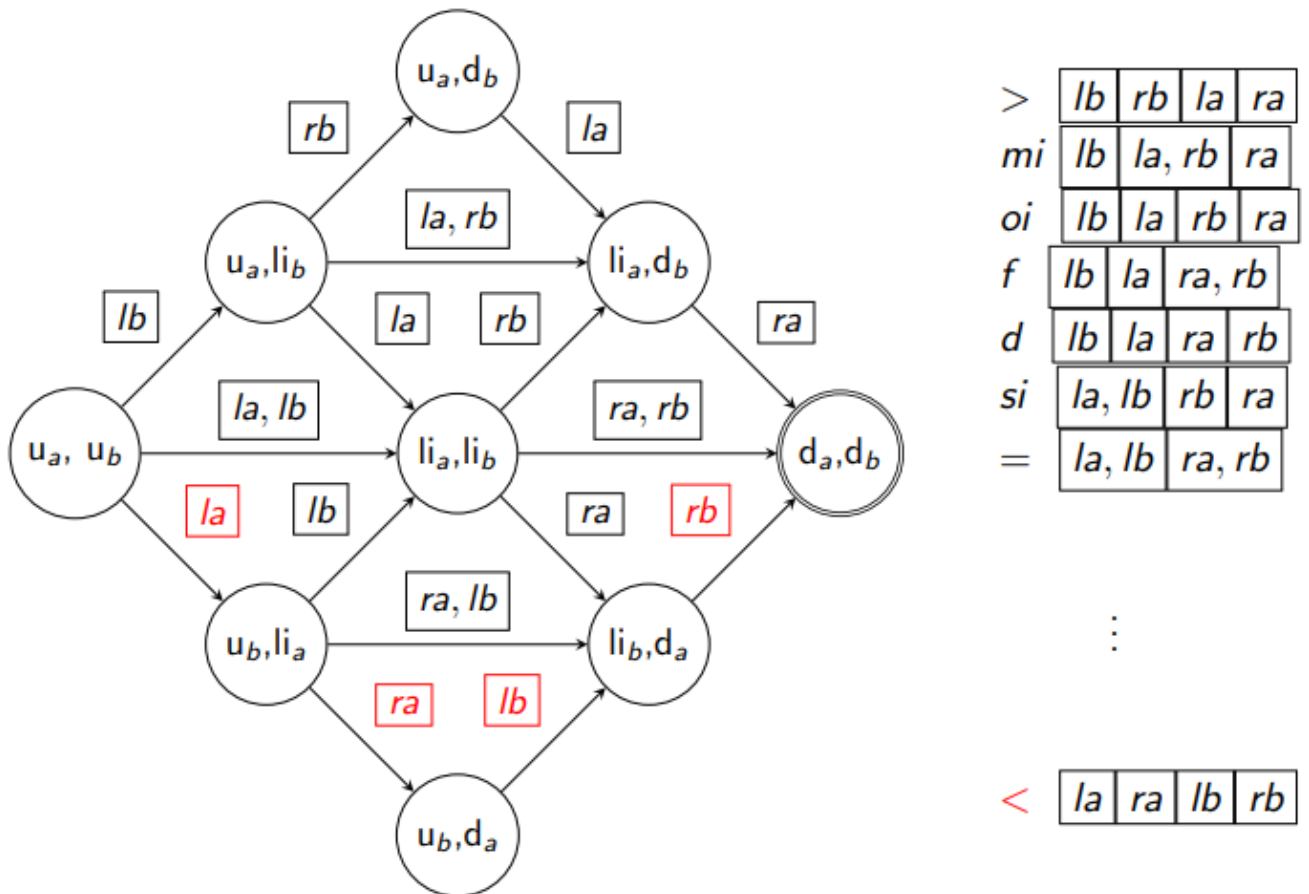


Fig 1.2.1

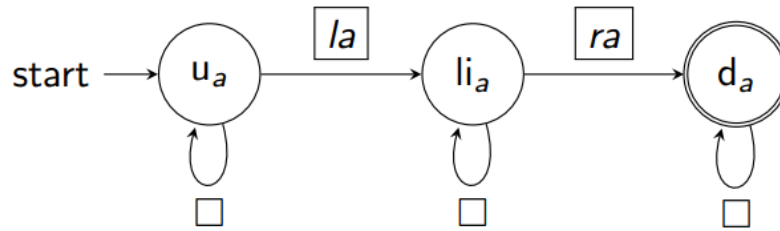
There are 13 possible Allen relations between two events a and b. All the 13 relations and how they relate with the example shown in Fig 1.2.1 are shown in table 1.2.1. If the system becomes complex and there are more than 13 possible ways for a system to happen over a time period all the ways can be grouped into these 13 relations and it becomes a bit simpler to analyze. 13 Allen relations are very effective in analyzing a non-deterministic system, we can group the possibilities and eliminate the unwanted possibilities or analyze the possibilities we want.

Allen relation	Symbol	Representation	Meaning
a Before b	b or <	la, ra, lb, rb	a is born and dies before b is born.
b Before a	bi or >	lb, rb, la, ra	b is born and dies before a is born.
a meets b	m	la, lb ra, rb	a is born before b but a die at same time at which b is born.
b meets a	mi	lb, la rb, ra	b is born before a but b dies at same time at which a is born.
a overlaps b	o	la, lb, ra, rb	a is born before b and a also dies before b dies
b overlaps a	oi	lb, la, rb, ra	b is born before a and b also dies before a dies
a starts b	s	la lb, ra, rb	Both a and b are born at same time but a dies before b
b starts a	si	la lb, rb, ra	Both a and b are born at same time but b dies before a
a during b	d	lb, la, ra, rb	b is born before a but a dies before b dies
b during a	di	la, lb, rb, ra	a is born before b but b dies before a dies
a finishes b	f	lb, la, ra rb	b is born before a but both a and b die at the same time.
b finishes a	fi	la, lb, ra rb	a is born before b but both a and b die at the same time.
a equal b	=	la lb, ra rb	Both a and b are born and dies at the same time.

Table 1.2.1

1.3 Superposing automata

To superpose we have to assume that at any time period an event has an option either for action to happen or nothing happens. So, in case of event a initially a is unborn i.e., u_a it has two options at every point of time period that it can either remain in unborn state i.e., do nothing that is represented by empty box in figure 1.3.1 or an action of a being born can happen i.e., l_a . Similarly, when a moves to a living state i.e., li_a it has two options at every point of time period that it can either remain in living state or an action of a dies can happen. Once a dies it we assume a do nothing till the end of all the time periods i.e., represented by a empty box at d_a state. Now, there are countable infinite many ways in which event a can happen independently same can be done for b resulting in infinite many ways in which event b can happen independently. For superposition we first do a Componentwise union between strings of the same length can they also can be grouped into the 13 Allen relations like we did in figure 1.3.1. A string from set of possible ways event a can happen (), l_a , ra and a string from a set of possible ways event b can happen l_b , rb , () (' ()' signifies empty set i.e., nothing happen in that time period), both string are of equal length which after componentwise union gives us l_b , l_a rb , ra which is b meets a. Once union is done, we depad the componentwise union 's' i.e., deleting all the empty sets from time period where nothing happened i.e., we can simply skip where either in event a or in event b nothing happens.



$$\square^* l_a \square^* r_a \square^* = l_a r_a + \square l_a r_a + \dots$$

Componentwise union \sqcup between strings of the same length

$$\begin{aligned} l_a r_a \sqcup l_b r_b &= l_a, l_b \mid r_a, r_b \\ \square l_a r_a \sqcup l_b r_b \square &= l_b \mid l_a, r_b \mid r_a \\ &\vdots \end{aligned} \quad \begin{aligned} &= \\ &mi \end{aligned}$$

After \sqcup , undo \square -loop insertion

$$depad(s) := s \text{ with } \square \text{ deleted}$$

Figure 1.3.1

For Superposing automata (over an alphabet of sets) we can say the following,

Given finite automata $M = \langle \rightarrow, F, q_0 \rangle$

$M' = \langle \rightarrow', F', q'_0 \rangle$

let $M \& M'$ be $\langle \rightsquigarrow, F \times F', (q_0, q'_0) \rangle$ with transitions \rightsquigarrow generated alongside a set P of pairs of states including (q_0, q'_0) as follows

$$(cu) \quad \frac{P(q, q') \quad q \xrightarrow{\alpha} r \quad q' \xrightarrow{\alpha'} r' \quad \alpha \cap A' \subseteq \alpha' \quad \alpha' \cap A \subseteq \alpha}{P(r, r') \quad (q, q') \xrightarrow{\alpha \cup \alpha'} (r, r')}$$

$$(d1) \quad \frac{P(q, q') \quad q \xrightarrow{\alpha} r \quad \alpha \cap A' = \square}{P(r, q') \quad (q, q') \xrightarrow{\alpha} (r, q')}$$

$$(d2) \quad \frac{P(q, q') \quad q' \xrightarrow{\alpha'} r' \quad \alpha' \cap A = \square}{P(q, r') \quad (q, q') \xrightarrow{\alpha'} (q, r')}$$

(d1) and (d2) follow from componentwise union (cu) assuming

$$q \xrightarrow{\square} q \quad \text{and} \quad q' \xrightarrow{\square} q'.$$

For Superposing languages (over an alphabet of sets) we can say the following,

Given a set X , an *S-string over X* is a string s of non-empty subsets of X .

Let $\mathcal{S}(X)$ be the set of *S-strings* over X

$$\mathcal{S}(X) = \{ \text{depad}(s) \mid s \in (2^X)^* \}.$$

Assuming M and M' accept *S-strings* over A and A' respectively

$$\mathcal{L}(M) \subseteq \mathcal{S}(A)$$

$$\mathcal{L}(M') \subseteq \mathcal{S}(A')$$

the language $\mathcal{L}(M \& M')$ accepted by $M \& M'$ consists of *S-strings* s over $A \cup A'$ such that

$$\text{depad}(\rho_A(s)) \in \mathcal{L}(M) \quad \text{and} \quad \text{depad}(\rho_{A'}(s)) \in \mathcal{L}(M')$$

where $\rho_X(s)$ is s intersected componentwise with X

$$\rho_X(\alpha_1 \cdots \alpha_n) := (\alpha_1 \cap X) \cdots (\alpha_n \cap X).$$

1.4 Python code for Superposition

The python libraries I studied and used for applying the superposition theory to make non-deterministic finite automata are **automata library** for creating a NFA (non-deterministic finite automata) given the states, input_symbols and transitions from one state to another and which symbols makes that transitions happen, **visual_automata library** so that I could show the NFA created, Figure 1.4.1 shows an example how I used these automata and visual_automata library further the output of **new_nfa.show_diagram()** can be seen in Figure 1.4.3 and lastly **itertools library** just for making all the possible combinations of input symbols Figure 1.4.2 shows what I used **itertools.combination()** function for.

```
import import_ipynb

from automata.fa.nfa import NFA
from visual_automata.fa.nfa import VisualNFA


new_nfa = VisualNFA(
    states = state,
    input_symbols = symbol,
    transitions = trans,
    initial_state= p,
    final_states= {'d d d '}
)

new_nfa.show_diagram()
```

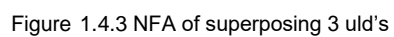
Fig 1.4.1

```
import itertools
```

```
for i in range(2,n+1) :
    b = list(itertools.combinations(r,i))
```

Figure 1.4.2

I was provided with a python script supFA.py which was a algorithm superposing n uld's(uld is a event shown in Fgure1.1.1), n being a whole number starting from 2 but the data was in not the form we can analyze or make a NFA out of it which was obtained from superposing through supFA.py so, I had to modify the data according to the requirement of NFA. I took the results obtained from supFA.py like stated, labels and trans and changed them to states such as 'u u u' , 'u l u' , etc. for n = 3 ; labels to input symbols which are combinations of -1 signifying an action of event 1 dying i.e., from living(l) state to dead (d) state , -2 signifying an action of event 2 dying , -3 signifying an action of event 3 dying , 1 signifying an action of state 1 being born i.e., from unborn (u) state to living (l) state , 2 signifying an action of event 2 being born , 3 signifying an action of event 3 being born and if nothing happens for an event that is signified by showing nothing for that particular event in a transition from one state to another so, an transition can be combination of these 6 action and nothing if nothing happens in an event for example,[-1, 2] means event 1 is dying event 2 is being born and nothing happens for event 3 ; and last but not least transitions I had to modify according to our NFA. After getting all the state , input symbols and transition I had to just call visualNFA function of automata library to make a NFA and lastly to show the diagram I used show_digram() function as done in Figure 1.4.1. The end result is an uld for any n Figure 1.4.3 shows a uld for n=3 . I tried to work to for n= 10 it works but takes a lot of time like 10's of minutes it also I think can work for any n so, the code is very robust.



2 Action languages

We saw in last sections that once events are superposed, they become complex and non-deterministic in nature so, to analyze effects of actions in non-deterministic events we need a formal model that complements well with 13 Allen relations. Action languages are those formal models for describing the effects of actions. This definition is taken from the Action languages paper by Michael Gelfond and Vladimir Lifschitz published in 1998 in Linköping Electronic Articles in Computer and Information Science. I have used Definitions of action signatures and the concept of transition system from this paper.

2.1 Action Signatures

Definition 1 *An action signature consists of three nonempty sets: a set V of value names, a set F of fluent names, and a set A of action names.*

Any Fluent has a specific value during a given time period in a life time of an event. Value of a fluent change over a life time of an event. An action brings about a change from one state to a new resulting state and states can be defined as all combination of fluents with all possible values. A state cannot be defined from its initial state and action as you can see in Fig.2.4.3 which make our action signatures non-deterministic in nature so, nothing can be assumed about a state in an action signature. Fig 1.1.2 is an action signature with fluents as 'a' and 'b' fluents ; values as 'u' , 'l' and 'd' and actions such as ' la, ra, lb, rb ' ,etc.

2.2 Transition Systems

Definition 2 A *transition system* of an action signature $\langle \mathbf{V}, \mathbf{F}, \mathbf{A} \rangle$ consists of

- (i) a set S ,
- (ii) a function V from $\mathbf{F} \times S$ into \mathbf{V} , and
- (iii) a subset R of $S \times \mathbf{A} \times S$.

A set of S is all the possible states that can exist in an action signature. A set A which is a set of actions. A function that maps one pair of states and actions to another pair of states and actions. We have compared a Transition System with a labeled directed graph. The initial state is denoted by an arrow as you will see in figure 2.4.1 and figure 3.2.1. the transition system represents all the possible transitions that can occur in the system, given the actions and starting state.

2.3 Elementary elements

To discuss the concurrent execution of actions in the transition system framework, we consider an action signature whose action names are truth-valued functions defined on a set \mathbf{E} of “elementary action names”:

$$\mathbf{A} = \{f, t\}^{\mathbf{E}}. \quad (1)$$

In action languages, elementary actions form the basis of action descriptions. An elementary action is the most basic action that has an effect on the state of the system being modeled. In our case of where we are considering bounding life of two events a and b and superposing them which are represented by a directed graph in Fig. 2.2.1 Elementary action can be represented

as we go from one state to another for example, $(u_a, u_b) \xrightarrow{l_a, l_b} (li_a, li_b)$ is a Elementary action which represents action of both a and b being born at the same time. We combine different elementary actions to form more complex actions for example in Fig. 2.2.1 we can represent the actions of bounding life of both a and b where both a and b are being born and die at the same periods concurrently $(u_a, u_b) \xrightarrow{l_a, l_b \mid r_a, r_b} (d_a, d_b)$ here we have combined two elementary action i.e., $(u_a, u_b) \xrightarrow{l_a, l_b} (li_a, li_b)$ and $(li_a, li_b) \xrightarrow{r_a, r_b} (d_a, d_b)$.

Elementary actions are now allowing me to use action languages in a manner in which I can represent a wide range of actions and systems. Elementary actions are also allowing me to keep a simple and expressive syntax. Defining more complex actions from basic elementary actions gives us a possibility to specify and reason a large number of actions, even when the systems are highly complex.

2.4 Python code for Action languages

I wrote a python script to study and use action languages. This is an action signature for 3 actions and fluents with values 2,5 and 2 which gives me a total state of $2*5*2$ i.e., 20 states with initial state as 0 [0,0,0] and final state as 15 [0,0,0]. I also introduced a transition i.e., $0[0, 0, 0]: \{0: \{0 [0, 0, 0]\}\}$ which makes it a transition system. I have used **automata library** for creating a NFA (non-deterministic finite automata) given the states, input_symbols and transitions from one state to another and which symbols makes that transitions happen, **visual_automata library** so that I could show the NFA created, Figure 2.4.1.

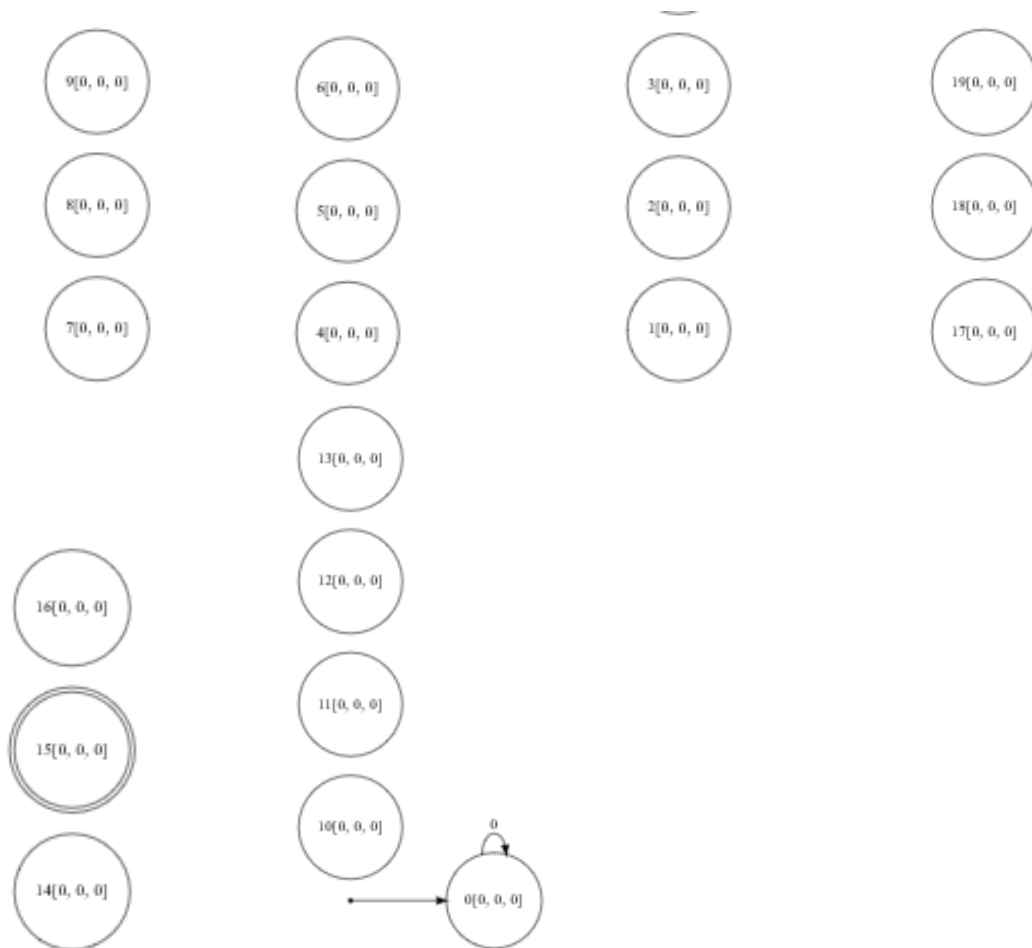


Figure 2.4.1 NFA of action signature signature for 3 actions and fluents with values 2,5 and 2.

```

n_nfa = VisualNFA(
    states = state,
    input_symbols = {0,1},
    transitions = {
        '0[0, 0, 0]' : {0 : {'0[0, 0, 0]'}}
    },
    initial_state = '0[0, 0, 0]',
    final_states = {'15[0, 0, 0]'}
)

n_nfa.show_diagram()

```

Figure 2.4.2

I was provided with two python scripts initList.py and actLang.py which were scripts to make a action signature but the data was in not the form we can analyze or make a NFA out of it which was obtained from initList.py and actLang.py scripts. so, I had to modify the data according to the requirement of NFA. I took the results obtained from the script like action names, fluents and value names and changed them to states, input_symbols , initial_state and final_state.

3. The Frame Problem for elementary actions

3.1 Bounding life of two events as an action language transition system.

Given finite non-empty set A , the set $3A$ represents the set of all possible function from A to set $\{0,1,2\}$ which are isomorphic to set the of triples (U,L,D) where U represents unborn, L represents living and D represents dead. The transition from one state to another in this ULD-automation are represented by action such as for example, $\{la, rb\}$.

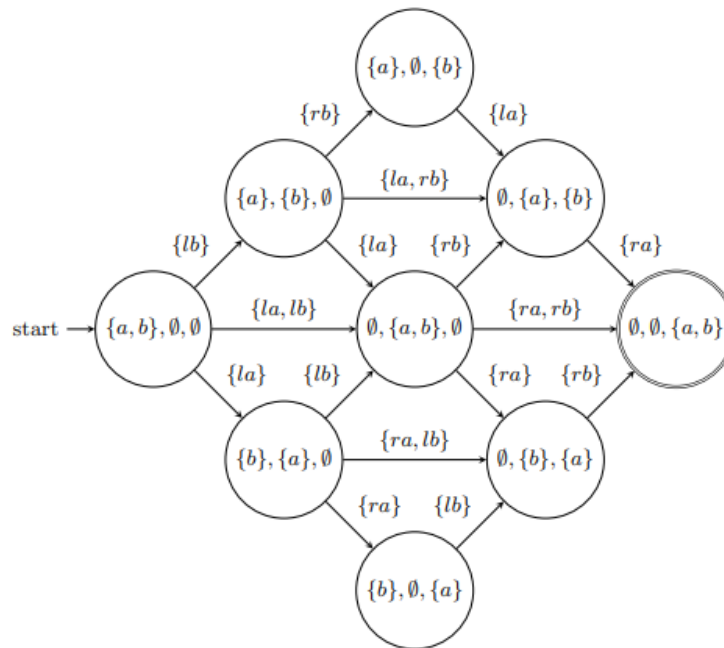


Figure 3.1.1

To model 'ULD' in the form of action languages we have to say that Fluent = person with values u, li, d (for unborn, living, dead). E.g. the U,L,D-state $(\{a\}, \{b\}, \emptyset)$ assigns a the value u , and b the value li i.e., $val(a, (\{a\}, \{b\}, \emptyset)) = u$ and $val(b, (\{a\}, \{b\}, \emptyset)) = li$ respectively.

The effects of elementary actions on the fluent variables in the ULD model is addressing how the values of fluent variables other than the one being changed by an elementary action are affected by that action. The inertial approach to frame problem we can assume there are no elementary actions that has an effect on an event in the system if that action does change the state of that particular event in a system, the elementary actions that has an effect on an event in the system are the ones who does make a change in the state of that particular event.

For example, for event 'a' here in fig 3.1.1 with a set of elementary actions as skip_a , la , ra . The effects of elementary actions are given by

$$q \xrightarrow{la} q' \implies \text{val}(a, q) = u \text{ and } \text{val}(a, q') = li$$

$$q \xrightarrow{ra} q' \implies \text{val}(a, q) = li \text{ and } \text{val}(a, q') = d$$

The inertial approach to the Frame Problem assumes there is no other elementary action, making $\text{val}(a', q) = \text{val}(a', q')$, as in

$$q \xrightarrow{\text{skip}_a} q' \implies \text{val}(a, q) = \text{val}(a, q')$$

for non-effects.

3.2 Python code

I modeled the ULD-automation in the form of action languages I used both the sup.py script and the script in generated when I was studying action languages and represented action languages using automata library. Fig 3.2.1 is the result of that. This is an action signature with two events and 2 fluent with 3 fluent values. Now action being born on event '1' would be represented by '1' i.e., positive of that event number and action being dead on event '-1' would be represented by '-1' i.e., negative of that event number.

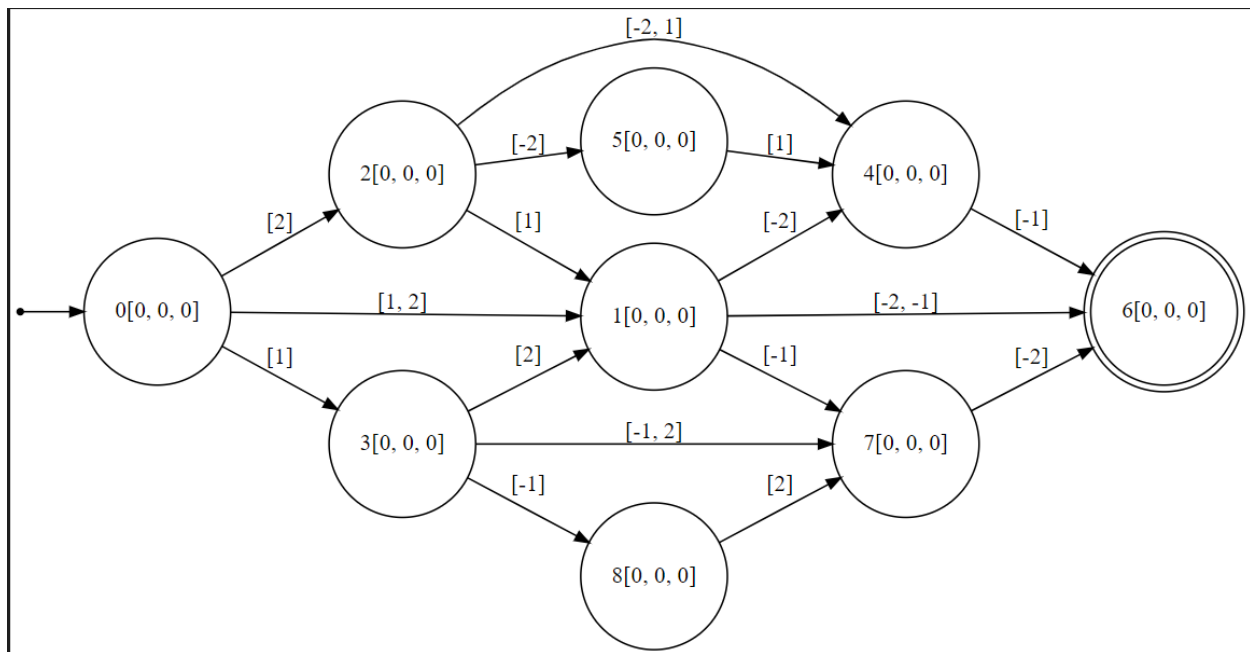


Figure 3.2.1

The libraries I used are automata library and visual_automata library to represent in a visual graph form . I also Itertools library just for making all the possible combinations of input symbols. Lastly I used string library.

```
from automata.fa.dfa import DFA
import actLang as al
import initList as iL
import string
import supFA as sup
import itertools
```

Figure 3.2.2

Security and privacy regarding Frame Problem

As I have no user data that I am using for this project so, I can't think of any privacy issues that can occur and for future system that counters the frame problem one thing is sure that those systems will be non-deterministic in nature so I have discussed some ways in which we can provide security to non-deterministic systems and later I have discussed some security and privacy issues regarding AI as frame problem is the biggest problem in AI, it is important to do so.

In my project I have used non-deterministic finite automata to represent a system with frame problem. The challenges in Encrypting NFAs are that they have an unbounded attribute lengths and unbounded key size. When I was researching for a method of encryption that can be implemented to non-deterministic finite automata, I came across this paper referenced below in references. In the paper they have constructed Attribute based Encryption for NFA with learning with error assumption, how the encryption works is if secret key associated to NFA is M of unbounded length, (x, m) is the tuple of cyphertexts where x is a public attribute of unbounded length and m is a secret message bit, and decryption recovers m if and only if $M(x) = 1$.

In 2012 Waters [Wat12] published a paper which is also referenced below provided the first Attribute based Encryption for Deterministic finite automata that supports unbounded attribute lengths and unbounded key requests which is one the most promising candidate for generalizing to NFA but it has seen no further progress. There are other works in the same area like Boyen and Li [BL15], Agarwal and Singh [AS17a] and Ananth and Fan [AF18] (all referenced below) all pose fundamental barriers towards generalization of encrypting NFA's. Attribute based Encryption for NFA with LWE assumption uses different techniques to handle unbounded length than other Encryption methods I discussed before. The encryption focuses on providing a secret symmetric key on Attribute based encryption scheme that supports unbounded length inputs but the NFA machine should be bounded size then "bootstrap" the construction of bounded size NFA machines to handle unbounded length machines.

So, in future if system or technologies are created on the concept of NFA then they can be securely encrypted by Attribute based encryption with learning with error assumption. Further the frame problem is a problem concerning AI so, the system that will be developed trying to minimize frame problem will be in AI so I will now discuss various security and privacy issues in AI in general and what can be done to make system in AI more secure.

The systems that are developed for AI they have vulnerabilities like other normal system but they may have addition opportunities for exploitation. Attackers are usually after the data-models or the data used for those data-models to work. AI systems are made secure keeping the data models and data for those data-models in mind. Few ways in which attackers can exploit AI systems are System Manipulation in which a system is provided with malicious inputs that does not exist in the system and causing the result to be false, Data Corruption and poisoning in which attackers can make the system behave in a manner that they want by corrupting or poisoning the data sets used by the systems and Transfer learning attacks in which attackers can attack the data models because of the predictable nature of the models, AI systems are developed and trained with having a specific purpose in mind which can be attacked.

Data privacy becomes very important in AI, not only the data generated or stored by AI systems but also the data sets, data inputs used by data models any leak can be very dangerous. Attackers try to launch inconspicuous data extraction attacks making the whole AI system at risk. One of the way organizations that uses or will use AI systems can protect themselves and also be open to people are set policies in place which would work to prevent these data extraction attacks.

The protection and prevention of AI systems becomes very important as stated due to reasons stated above, here are few ways in which we can secure AI systems- using good data hygiene this can be insured by having to use only necessary, accurate and fair data both while training and implementation the AI systems, and try to keep this training and implementation data as secure as possible and should be discarded and deleted as soon as possible, making robust algorithms this can be insured by trying pre-meditate and diversifying data for situations that can be faced during attacks and having a counter measure for that situation.

Conclusion

The conclusion I came up with is an action is a combination of various elementary actions, by this understanding it makes us easier to study very big and complex large models. One of the cases where the understanding of elementary actions is useful is that we can assume there are no elementary actions that has an effect on an event in the system if that action does change the state of that particular event in a system, the elementary actions that has an effect on an event in the system are the ones who does make a change in the state of that particular event.

Bibliography

All the work and paper were provided to me by professor Tim Fernando and here is the link of all the things he provided me with.

<https://www.scss.tcd.ie/~tfernand/FP/>

Security and Privacy concerning regarding frame problem

Shweta Agrawal, Monosij Maitra and Shota Yamada, *Attribute Based Encryption (and more) for Nondeterministic Finite Automata from LWE* Available at <https://eprint.iacr.org/2019/629.pdf>

[Wat12] Brent Waters. Functional encryption for regular languages. In Crypto, 2012.

[BL15] Xavier Boyen and Qinyi Li. Attribute-based encryption for finite automata from lwe. In ProvSec, 2015.

[AS17a] Shweta Agrawal and Ishaan Preet Singh. Reusable garbled deterministic finite automata from learning with errors. In ICALP, volume 80. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[AF18] Prabhanjan Ananth and Xiong Fan. Attribute based encryption with sublinear decryption from lwe. Cryptology ePrint Archive, Report 2018/273, 2018. <https://eprint.iacr.org/2018/273>.

<https://hubsecurity.com/blog/cyber-security/security-threats-for-ai-and-machine-learning/>