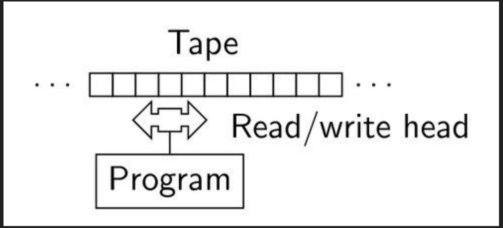
UTM Presentation 3

By Will Manley, Jason Giroux, and James Lee

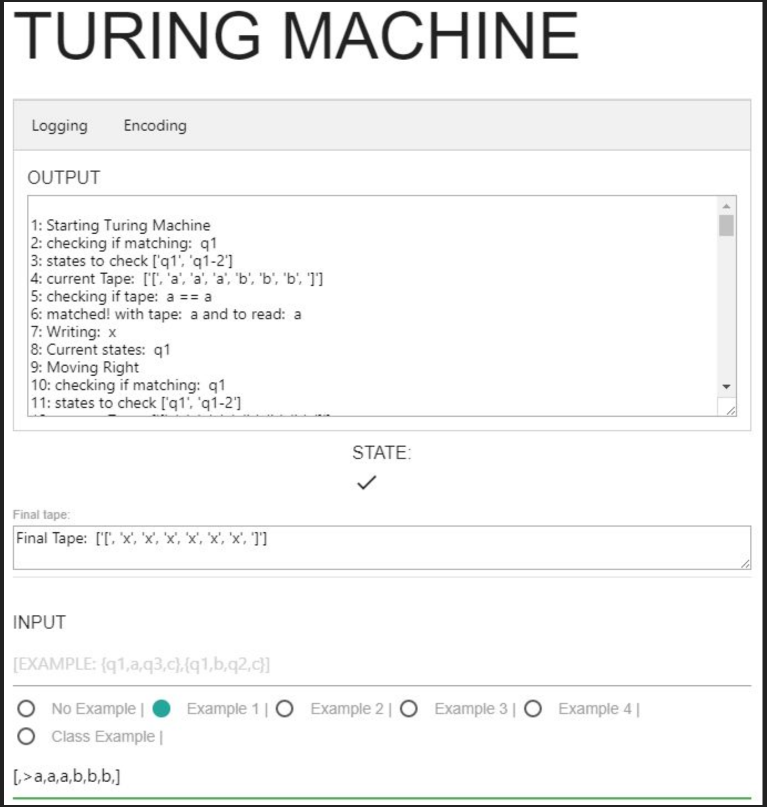
General Approach

- Using electron (frontend) and Python (backend)
- Machine will prompt user to input a tape and transition information
- Parse transitions into JSON file
- Print each step of the machine
- Output the final stage of the machine



Input / Transitions

- Input alphabet {a,b,c,...}
  - Final version supports all characters of the alphabet, deltas, and brackets.
- Tape alphabet {a,b,c,X,Y,>,∆,[,]}
- Will prompt user to enter transitions with the following format:
  - {q1,a,q2,b,R},{q1,b,q3,c,R}...
  - Or choose an example that are defined in the “directions” page of the UI
- UI has options for examples users can use to learn about Turing machines without writing their own transitions



```
1: Starting Turing Machine
2: checking if matching:  q1
3: states to check ['q1', 'q1-2']
4: current Tape:  ['[', 'a', 'a', 'a', 'b', 'b', 'b', ']']
5: checking if tape:  a == a
6: matched! with tape:  a and to read:  a
7: Writing:  x
8: Current states:  q1
9: Moving Right
10: checking if matching:  q1
11: states to check ['q1', 'q1-2']
```

STATE:
✓

Final tape:

```
Final Tape:  ['[', 'x', 'x', 'x', 'x', 'x', 'x', ']']
```

INPUT

[EXAMPLE: {q1,a,q3,c},{q1,b,q2,c}]

No Example | [✓ Example 1] | Example 2 | Example 3 | Example 4 | Class Example |

[,>a,a,a,b,b,b,]

Internal Data structures

JS - Input: parses input, calls first python socket to create JSON datafile depending on the example number value.

JS - DisplayRatioValue: Gets the example number if any are checked

Python - ReadJSON: reads the JSON file that was written to.

Python - readTape: Tape data is first stored in a txt file, then processes in python after Node parses it.

Python - TapePadding: Adds padding to the sides of the tape if they are unbounded

Python - startingPosTape: returns where the “>” is on the tape

Python - Encode: parses through the transition JSON file and encodes it with a trailing 0 and library corresponding to the number of 1’s

Python - UTM: main function for the turing machine

Python - whatToRead: Since everything is object oriented, this function returns the attributes of the current object.

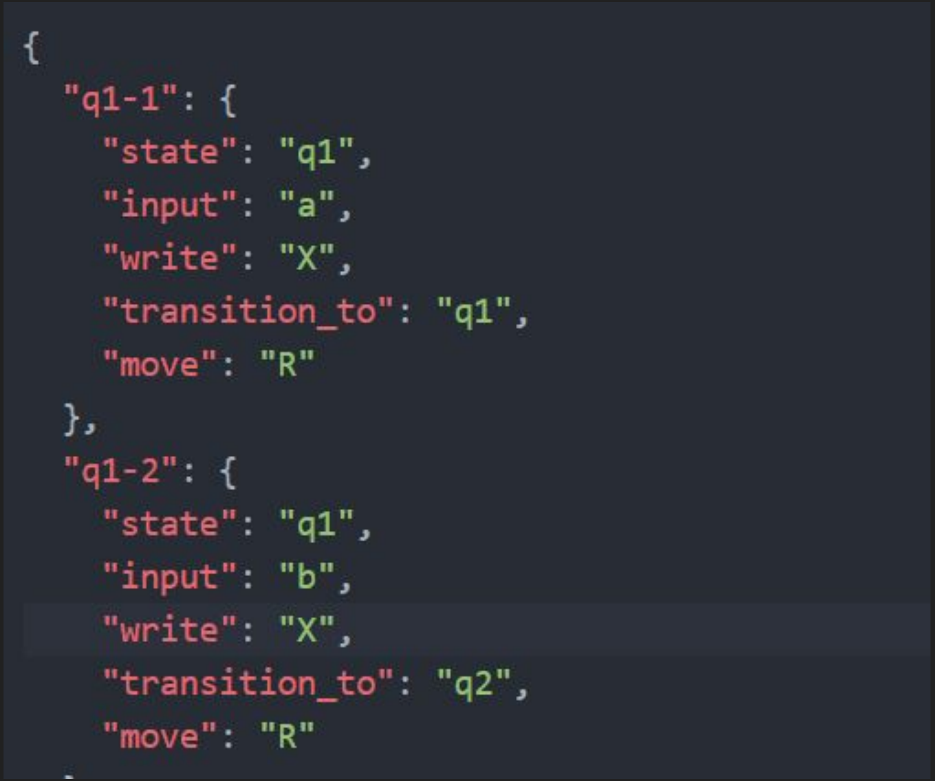
Handling Transitions

Transitions will be stored in a JSON file that will be formatted by the front end.

The JSON file will be populated by the user input.

Format:

- State
- Input
- Write
- Transition state
- Move



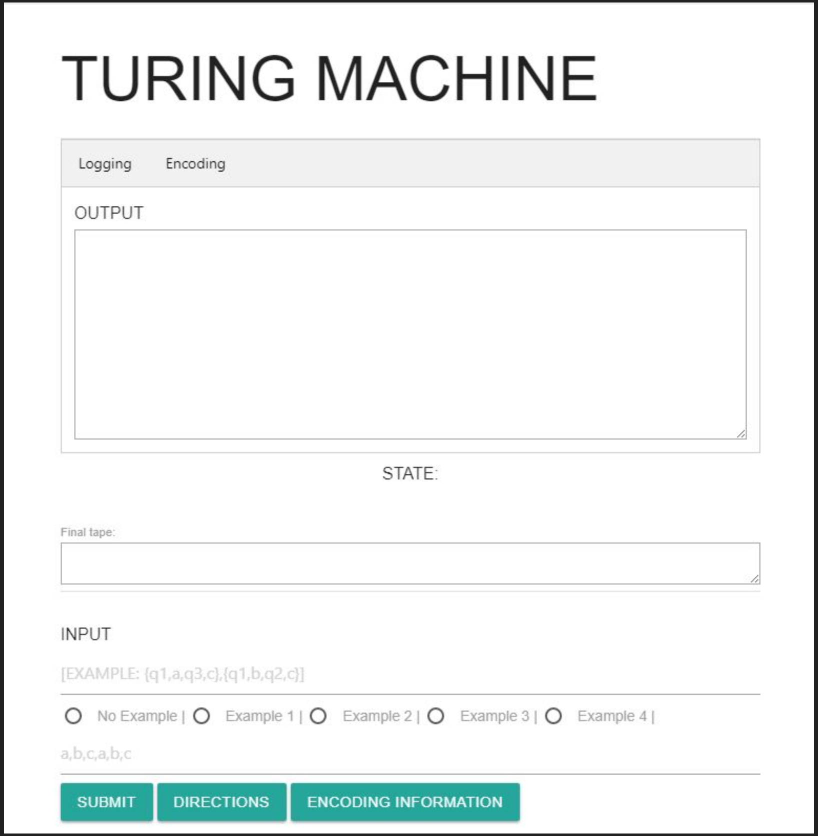
```
{
  "q1-1": {
    "state": "q1",
    "input": "a",
    "write": "X",
    "transition_to": "q1",
    "move": "R"
  },
  "q1-2": {
    "state": "q1",
    "input": "b",
    "write": "X",
    "transition_to": "q2",
    "move": "R"
```

Deployment

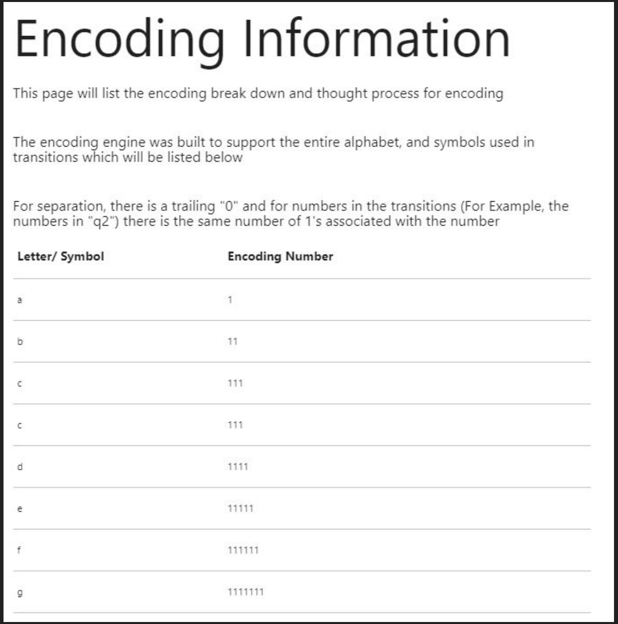
- Cross platform
  - Confirmed to work on:
    - MAC OSX
    - Windows 10
- Pull from the Github Repo
  - Install Node JS dependencies
    - “Npm i”
  - Install python
- Start program:
  - Npm start

Finished UI

- Electron front-end
- **State**: Displays a ***check*** or ***X*** if accepted or rejected
- **Output**: shows traceability and final tape
- **Input**: User inputs desired input
- Python socket for backend communication
- UI will be responsible for parsing the user input to determine if it is the right syntax for the backend
- Backend processes the parsed input and returns the traced stack, final tape, encoding, and state



Directions & Encoding



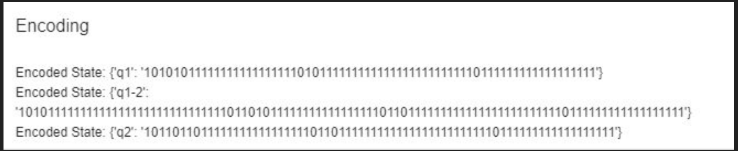
Encoding Information

This page will list the encoding break down and thought process for encoding

The encoding engine was built to support the entire alphabet, and symbols used in transitions which will be listed below

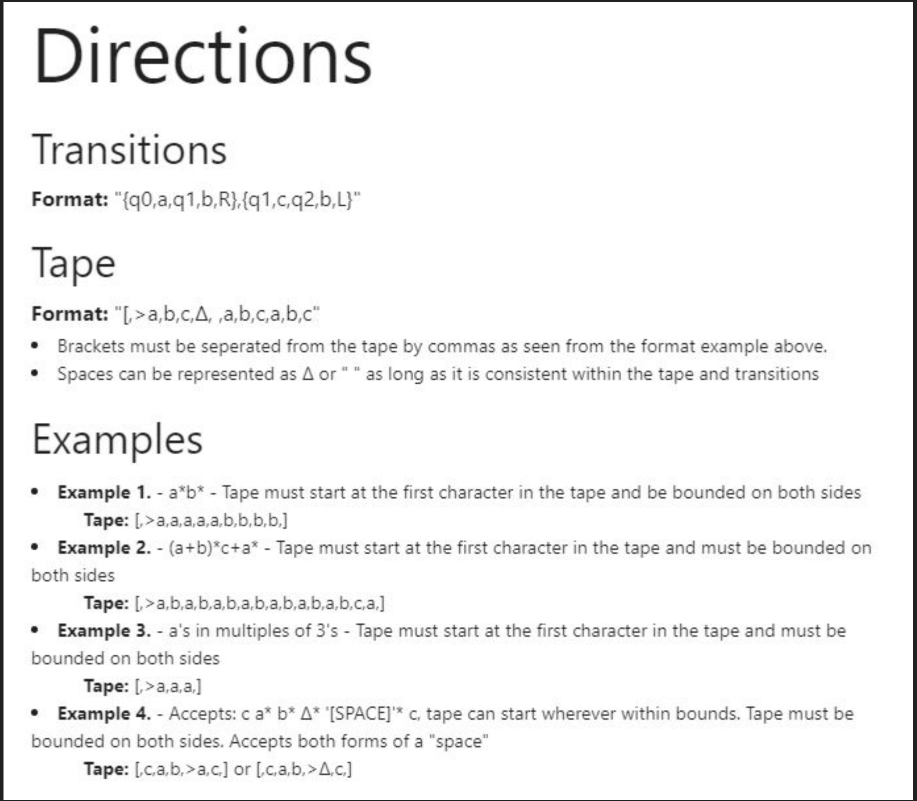
For separation, there is a trailing "0" and for numbers in the transitions (For Example, the numbers in "q2") there is the same number of 1's associated with the number

Letter/ Symbol	Encoding Number
a	1
b	11
c	111
c	111
d	1111
e	11111
f	111111
g	1111111



Encoding

Encoded State: {'q1': '10101011111111111111111101011111111111111111111111110111111111111111111'}
Encoded State: {'q1-2':
'1010111111111111111111111111111101101011111111111111111011011111111111111111111111110111111111111111111'}
Encoded State: {'q2': '10110110111111111111111110110111111111111111111111111101111111111111111111'}



Directions

Transitions

Format: "{q0,a,q1,b,R},{q1,c,q2,b,L}"

Tape

Format: "[,>a,b,c,Δ, ,a,b,c,a,b,c"

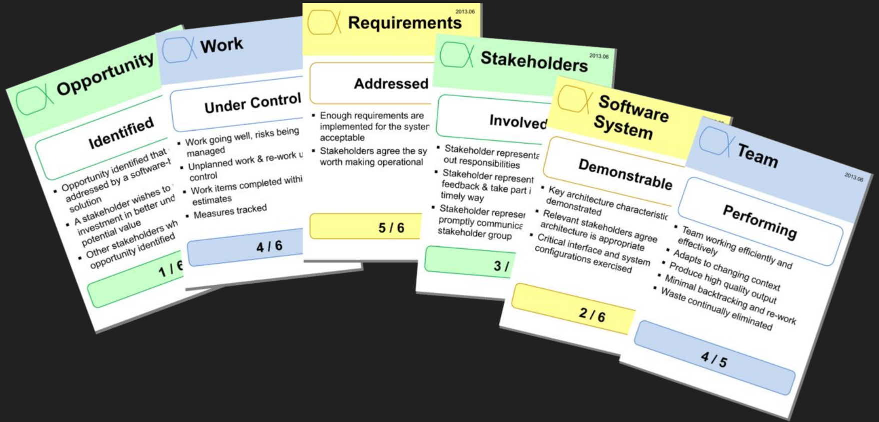
- Brackets must be seperated from the tape by commas as seen from the format example above.
- Spaces can be represented as Δ or " " as long as it is consistent within the tape and transitions

Examples

- **Example 1.** - a*b* - Tape must start at the first character in the tape and be bounded on both sides
  Tape: [,>a,a,a,a,a,b,b,b,b,]
- **Example 2.** - (a+b)*c+a* - Tape must start at the first character in the tape and must be bounded on both sides
  Tape: [,>a,b,a,b,a,b,a,b,a,b,a,b,a,b,c,a,]
- **Example 3.** - a's in multiples of 3's - Tape must start at the first character in the tape and must be bounded on both sides
  Tape: [,>a,a,a,]
- **Example 4.** - Accepts: c a* b* Δ* '[SPACE]'* c, tape can start wherever within bounds. Tape must be bounded on both sides. Accepts both forms of a "space"
  Tape: [,c,a,b,>a,c,] or [,c,a,b,>Δ,c,]

Essence Cards

- We have completed 99% of the essence cards before the “Software System” cards.
- Remaining card(s):
    - Opportunity
        - Stakeholders satisfied
    - Way of Working
        - Retired card


Opportunity
Identified
Opportunity identified that addressed by a software-b solution
A stakeholder wishes to investment in better und potential value
Other stakeholders wh opportunity identified
1 / 6
Work
Under Control
Work going well, risks being managed
Unplanned work & re-work u control
Work items completed withi estimates
Measures tracked
4 / 6
Requirements
2013.06
Addressed
Enough requirements are implemented for the syster acceptable
Stakeholders agree the sy worth making operational
5 / 6
Stakeholders
2013.06
Involved
Stakeholder representa out responsibilities
Stakeholder represent feedback & take part i timely way
Stakeholder represer promptly communica stakeholder group
3 /
Software System
Demonstrable
Key architecture characteristic demonstrated
Relevant stakeholders agree architecture is appropriate
Critical interface and system configurations exercised
2 / 6
Team
2013.06
Performing
Team working efficiently and effectively
Adapts to changing context
Produce high quality output
Minimal backtracking and re-work
Waste continually eliminated
4 / 5

Demonstration:

Questions?