3 March 2015

Jakub Ciecierski

Cellular automaton
Requirement specification

Contents

1 Schedule

Date	Asset
2015-04-02	Technical project
2015-04-23	Code of modules
2015-04-30	version 0.98
2015-05-07	version 0.99
2015-05-14	version 1.00
2015-05-28	Test report
2015-06-11	Acceptation

2 Document metric

Document metric					
Project:	Cellular Automaton	Company:	WUT		
Name:	Requirement specification				
Topics:	Business analysis of the product				
Author:	Jakub Ciecierski				
File:	requirement_specification.pdf				
Version no:	0.1	Status:	Under development	Opening date:	2015-03-03
Summary:	Business analysis of application that allows for creating a cellular automaton				
Authorized by:	Wadysaw Homenda Lucjan Stapp	Last modification date:			2015-03-03

3 History of changes

History of Changes			
Version	Date	Who	Description
0.1	2015-03-03	Jakub Ciecierski	Definition of the main purpose of the document

4 Glossary

- **Pattern Recognition** In broad terms, pattern recognition is science of making assumptions about data using various tools from statistics, machine learning and many others fields. Focuses on designing and building machines that can recognize patterns. Such patterns can be found in speech, fingerprint, optical characters etc.

  Feature is defined as a quality or characteristic of an element. Such feature can be a symbolic measure (e.g. color) or numeric (e.g. width). Collection of d features is called a d-dimensional feature vector.

  In classification a pattern can be represented by a pair (x, w) where x is the feature vector and w is label. A label tells the computer to which class a given element belongs to. Elements from the same class should have similar features, while elements belonging to different classes should have relatively different features.

- **Alphabet** Is a finite, non empty set of symbols, commonly denoted by Σ. Examples of common alphabets:

  1. $\Sigma = \{0, 1\}$ - binary alphabet.
  2. $\Sigma = \{a, b, ..., z\}$ - small letters of latin alphabet.

- **Word over Alphabet** Also called a *string*, is a sequence of symbols over some alphabet Σ. Examples of words:

  1. A sequence $'01010'$ is a word over binary alphabet $\Sigma = \{0, 1\}$.
  2. A word $'lorem'$ is a word over the latin alphabet $\Sigma = \{a, b, ..., z\}$.

  An empty word is a word with no symbols. Commonly denoted by ε. Such word can be taken from any alphabet.

  A set of all words over alphabet Σ is denoted by Σ^*, where

  $$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ... \tag{1}$$

  If $\Sigma = \{0, 1\}$ then $\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \{0, 1\}$, $\Sigma^2 = \{00, 01, 10, 11\}$, $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$ and so on

  It is important to note that, Σ^* is infinite countable set.

- **Language** Language over alphabet Σ will be denoted by L. Language L is a subset of all words Σ^*, $L \subseteq \Sigma^*$.

  Examples of languages:

  1. $L = \{\varepsilon, 01, 10, 0011, 0101, 0110, ...\}$ - a set of all binary words that have the same number of occurrences of $0's$ and $1's$
  2. $L = \emptyset$ - language is an empty set, contains no words
  3. $L = \{\varepsilon\}$ - language containing only empty word.

- **Deterministic Finite Automaton (DFA)** Automaton is a very simply computability model. It can be thought of as a physical machine containing a *tape* with input word, a *head* reading a single symbol from the tape and finally a steering mechanism which can change its state during the computations based on current state and a symbol that is being read. DFA computes a word in order to check if a given word belongs to a language accepted by this machine.

  Formally DFA is a system of five fields:

$$A = (Q, \Sigma, \delta, q_0, F) \tag{2}$$

  where
  Q - finite set of states.
  Σ - Finite input alphabet.
  δ - transition function. $\delta : Q \times \Sigma \to Q$
  q_0 - the initial state. $q_0 \in Q$
  F - Set of accepting states. $F \subseteq Q$

  The computations of DFA is a sequence of transitions based on transition function. Depending on a state q and symbol x read by the head the machine:

  1. changes its state to $p \in Q$
  2. moves the head one cell to the right.

  Automaton finished computations when all symbols were read. It accepts input if computations end in accepting state, otherwise the input is rejected.

5 Goal

The main goal of this project is to deploy application, which will create an automaton for given input data. Produced automaton will be a an accurate classifier of objects represented by the input data. The program is dedicated to reasearch laboratory, hence it is lumbered with the following assuptions.

First of all, all users will be scientists, so precision of calculations and reliability is vital. We want to be sure about results given by the application to such an extent, that they will be publishable. It is also carrying need for specific format of the output - by default latex tables and .xls files. Similarly input is in form of .xls files.

Next thing that we want to stress out is platform and design. As for target system, linux is unquestionable choice. All work stations are running Arch Linux and we want the program to be operable on all of them. Although most of the researchers work inside the laboratory, some of us are using SSH protocol to communicate. This causes the need for plain console application - configurable using flags or simple question/answer scheme.

But we do not want to limit ourselves only to this approach - finally vast majority of us use computers via the standard X Window System and want to benefit from it. For those who does, we want to present simple GUI based program to configure, run and monitor the process of calculating automata. It will have all functionalites of console part, but will be more easy on the eye and simpler to use for non computer scientist.

Last but not least, we will tackle resources consuption and critical situation handling. On this point let us be clear: we want accurate results - neither time nor memory are important. The assumtion of course holds to some reasonable extent - we do not want to wait a month for program's output, but we are rather used to wait for couple of days. Great solution in this case would be ability to adjust complexity of calculations and, what follows, time needed to complete. With such an estimation, we could easily schedule our work.

6 User stories

6.0.1 GUI

As a user I want to:

- load data using file explorer.
- load data using drag and drop procedure.
- adjust computation precision and see estimated time to complete.
- select output format as .xls file.
- select output format as latex table.
- select destination folder of the output
- decide if test should be rerun in case of failure/interruption.
- start computation for loaded data.
- stop specific computation.
- stop all computations.
- monitor number of currently running computations
- monitor estimated time of all computations
- monitor progress of single computation.
- close application at any time
- minimize application at any time.
- resize application window.

7 Functional Requirements

In all tables of the following section we assume that priority can take following values:

- 1 - must be implemented
- 2 - can be implemented optionally
- 3 - is a nice addition, but not needed.

7.1 GUI

ID	Requirement	Comments	Priority
1	The system provides option to load data file using buttun 'Load data' and some window explorer to choose a file		1
1.1	The system provides area over which one can drag and drop data file. Following action will have similar results to requirement of id 1.		1
2	When new data is loaded, one can adjust computation precision via various checkboxes, sliders etc. Implementation of this part will depend on used algorithm.		1
2.1	Time estimation indicator, dynamically updated during usage of items from requirement of id 2		1
3	Application will have expanded list with possible output files. Choosing one will affect a way of saving result.		1
3.1	Option of .xls file on list described in 3		1
3.2	Option of latex table on list described in 3		1

4	System will provide 'Choose Output Folder' button which will open window explorer and ask user to choose folder to save result from a test.		1
4.1	Label with path to the output folder, chosen in requirement number 4.		2
5	Opportunity to choose behaviour after crashing of a particular test - rerun or not		1
6	Button 'Start Computation', which will begin computing automaton for a loaded data.		1
7	'Stop Computation' button, for each currently running computation. It will cause particular computation to break whatever it is doing right now.		1
8	'Stop All Computations' button. All computations break whatever they are doing right now.		1
9	System will provide a way of monitoring number of currently running computations. It can be in form of a label or some bar.		1
9.1	System will provide a way of monitoring estimated time of all currently running computations. It can be in form of a label or some bar.		1
9.2	System will provide a way of monitoring estimated time of a particular computation. It can be in form of a label or some bar.		1

10	By clicking some superior button (like e.g. 'X') user will be able to immediately close the application and terminate all computations.		1
10.1	By clicking some superior button user will be able to minimize the program.		2
10.2	By grabbing edges user will be able to resize application window.		3

7.2 Console

8 Non Functional Requirements