Finite Automata Models in Agro-ecosystem and Plant Protection

Svetla Maneva
Institute of Soil Science,
Agrotechnologies and Plant Protection,
Dept. of Plant Protection - Fythopathology
2230 Kostinbrod, Bulgaria

Krassimir Manev
New Bulgarian University,
Deptartment of Informatics
21, Montevideo street
1618 Sofia, Bulgaria

ABSTRACT

Diseases in plants cause major production and economic losses in agricultural industry worldwide. Monitoring of health and detection of diseases in plants and trees is critical for sustainable agriculture. In this paper an approach for building software systems for plant disease control that perform real time prediction of the outbreak and the development of the disease is proposed. Mathematical base of the approach are the finite automata. A method for transforming a given biological model of the disease to a finite automaton is developed. The software system has just to interpret the obtained automaton. The applicability of the developed approach is demonstrated on an example of a software system for prediction of development of *Phitophthora infestans* on potatoes and tomatoes. Predicting the breakout of the disease is very important for sustainable agriculture of Bulgaria because on favorable conditions the pathogen could destroy nearly 100% of the yield of potatoes and tomatoes.

General Terms:

Finite automaton, Plant disease control

Keywords:

Phitophthora infestans,Biological model, Real time prediction

1. INTRODUCTION

Diseases in plants cause major production and economic losses in agricultural industry worldwide. Monitoring of health and detection of diseases in plants and trees is critical for sustainable agriculture [10]. For the purpose of plant protection practice many chemical and biological products as well as methods for their application have been developed. However, if the control relies only on the application of expensive chemical treatment, then a correctly timed application of the spray, based on prediction of disease outbreaks, will be important in improving the effectiveness of the treatment and in reducing its cost.

A correct prediction of the disease occurrence ensures that the control measures, especially the application of chemical treatments or biological control agents, are used more effectively [1]. According these authors the plant disease forecasting uses weather data, frequently combined with biological data, to predict disease occurrence and incidence. Early information on crop health and disease detection can facilitate the control of diseases through proper management strategies [10].

In [11] it was reported the computer model Maryblyt, for predicting specific infection events and symptom development for most phases of fire blight epidemics (caused by Erwinia amylovora) in apples and pears. The program operates in real time to assess the current risks or the progress of an epidemic, or in a simulation mode for predicting future events using forecasted weather data. In [3] authors used weather data (temperature, rainfall, etc.) and mean disease levels collected in Ohio from 1982 to 1999 to identify critical environmental periods. In [4] statistical methods and advanced computer simulation are applied to predict and assess the risk of Asian soybean rust (Glycine max L. Merril), one of the most important fungal diseases of this crop.

For creating of a good disease development prediction system it is very important to base it on a strong statistical observations. As many as possible factors have to be underlined and the relationships among them to be established. Finally the space of all possible factor's values is divided in input categories—some of them favorable for the development of the disease, some of them—unfavorable. In [12] and [5] the authors used Bayesian analysis to evaluate the correctness of the conditional probability of pest occurrence.

This paper describes the fundamentals of a predicting software system that uses a known biological model. Because there is no essential information how the existing prediction systems are constructed we propose an universal approach for easy implementation of such system using a finite automaton.

2. FINITE AUTOMATA

Finite automata are *abstract mathematical machines* [2]. They provide a general algorithm which could be easy adapted for solving very different kinds of tasks—from processing texts, written in formal (or formalized in some way) languages, to solving tasks similar to the discussed below from the domain of plants protection. This permits creating of a relatively simple software that could solve all mentioned tasks in uniform way: emulating the work of the finite automaton which is modeling the task. For the purpose the task has

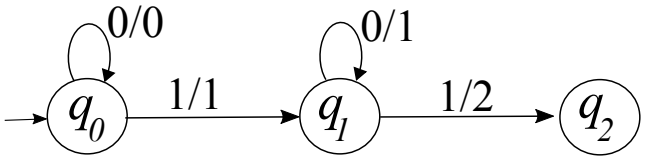

Fig. 1. Finite automaton.

to be "translated" to the language of the finite automata and the translation to be given to the emulating program as input (or to be embedded in it).

Difficulties of solving tasks with such approach are related only to modeling the corresponding task with an automaton. But the efforts to model some behavior with an abstract state machine (including finite automata) could be very helpful for the specialists in the domain of the solved task. Because the process of modeling usually leads to performing detailed analysis of the problem and clarification of all related details.

Abstract mathematical machines are formal computational tools that could solve tasks. Each such machine works in a *discrete time axe*—the nonnegative half of the real time axe on which only the integer moments of the time are chosen. Everything that the machine does happens only in the moments of the discrete time. The work done by the machine in each discrete moment of the time is called *step*.

Definition A *finite automaton* is a sextuple $A =< Q, X, Y, q_0, \delta, \lambda >$ where:

— Q is a finite set of *states* of the automaton. In each moment of the discrete time the automaton could be and has to be in a precisely one of its states. The states of the automaton are its history. Each state of the automaton "corresponds" to some important element of the process which is modeled by the automaton. It is possible that two different states correspond to the same element of the process. But this possibility could be eliminated and the automaton to be replaced by an equivalent, from the point of view of the solved task, such that different states correspond to different important moments of the process - *minimized* finite automaton.

— X is a finite set of *input characters* – the *input alphabet* of the automaton. With *words* of these characters we represent the data that we will introduce in the automaton. The set of words over X are denoted by X^*.

— Y is a finite set of *output characters* – the *output alphabet* of the automaton. With output characters from Y we represent the results of the work of the automaton. One of the output characters of the automaton could be the *blank* denoted by $\flat$. It could be used in moments when the automaton has noting to output.

— The state q_0 is called *initial*. The automaton always starts the work in the initial state, i.e. in the zero moment of the discrete time the automaton is always in the state q_0.

— The function $\delta : Q \times X \to Q$ is called *transition function* of the automaton. It is usually defined for each state $q \in Q$ and each character $x \in X$, and its values are states. If the state of the automaton in i-th moment of the discrete time is $q(i)$ and we give on the input the character x then in the next moment of the discrete time the automaton will pass into the state $q(i+1) = \delta(q(i), x)$. It is possible that the function δ is not total, i.e. it is not defined for each state $q \in Q$ and each $x \in X$. If $\delta(q(i), x)$ is not defined then the automaton stops its work.

— The function $\lambda : Q \times X \to Y$ is called *output function* of the automaton. It is usually defined for each state $q \in Q$ and each $c \in X$, and its values are output characters. If the state of the automaton in i-th moment of the discrete time is $q(i)$ and we give on the input the character x the automaton will show on the output the character $y = \lambda(q(i), x)$. It is possible that the function δ is not total also, i.e. it is not defined for each state $q \in Q$ and each $x \in X$. If $\lambda(q(i), x)$ is not defined then the automaton shows on the output the blank output character and continues its work.

Let us consider one example. Some system is a subject of damage within some bad conditions. It is known that the system could survive after one time interval of such conditions (let us call it "bad day") but after a second such interval it will be destroyed. We could model the behavior of the system with a finite automaton with 3 states with the following meaning:

— q_0 means that the system is in a **normal** state.

— q_1 means that the system survived **one bad day** and a second such period will destroy it.

— q_2 means that the system is **destroyed**.

The input alphabet of the automaton will be $X = \{0, 1\}$, where 0 means that the passed time period was with normal conditions (let us call it "good day"), and 1—that the passed time period was a bad day.

The output alphabet of the automaton will be $Y = \{0, 1, 2\}$, where 0 means that the state of the system is normal, 1—that the system is in a danger and 2—that the system is destroyed. For this example we do not need a blank output character.

So, $\delta(q_0, 0) = q_0$ which means that when the system is in a normal state and the day is good the system remains in a normal state. But $\delta(q_0, 1) = q_1$ which means that after a bad day the normal system pass into a state of danger. Analogically, $\delta(q_1, 0) = q_1$ and $\delta(q_1, 1) = q_2$, i.e. if while being in a state of danger the current day is good then the system stays in a state of danger. But when in a state of danger a bad day happens—then the system passes into the state destroyed. The values $\delta(q_2, 0)$ and $\delta(q_2, 1)$ are simply not defined because it is senseless.

The current state of the system will be shown on each step by the value of the output function. Thus $\lambda(q_0, 0) = 0$, $\lambda(q_0, 1) = \lambda(q_1, 0) = 1$, $\lambda(q_1, 1) = 2$, $\lambda(q_2, 0)$ and $\lambda(q_2, 1)$ are not defined.

On Fig. 1 the defined above finite automaton is shown graphically. The states are shown as circles. When $\delta(q', x) = q''$ then the circles of q' and q'' are linked with an arrow directed from q' to q'' and the character x is written near the arrow, followed by a slash character

('/') and the value $\lambda(q', x)$. The arrow that comes from no state and enters q_0 just marks the initial state.

Suppose that the time intervals are days and in a sequence of 8 days the corresponding conditions are coded with the sequence 00100011, i.e. after the first two days with favorable conditions a bad day follows. Then, after tree good days, we have finally two bad days again. Then, starting in q_0 the automaton will pass through the following sequence of states:

$$\begin{aligned} q(0) &= q_0 \\ q(1) &= \delta(q(0),0) = \delta(q_0,0) = q_0 \\ q(2) &= \delta(q(1),0) = \delta(q_0,0) = q_0 \\ q(3) &= \delta(q(2),1) = \delta(q_0,1) = q_1 \\ q(4) &= \delta(q(3),0) = \delta(q_1,0) = q_1 \\ q(5) &= \delta(q(4),0) = \delta(q_1,0) = q_1 \\ q(6) &= \delta(q(5),0) = \delta(q_1,0) = q_1 \\ q(7) &= \delta(q(6),1) = \delta(q_1,1) = q_2 \end{aligned}$$

and the automaton will stop because $\delta(q_2, 1)$ is not defined. As a result the automaton will produce the following sequence of outputs:

$$\begin{aligned} y(1) &= \lambda(q(0),0) = \lambda(q_0,0) = 0 \\ y(2) &= \lambda(q(1),0) = \lambda(q_0,0) = 0 \\ y(3) &= \lambda(q(2),1) = \lambda(q_0,1) = 1 \\ y(4) &= \lambda(q(3),0) = \lambda(q_1,0) = 1 \\ y(5) &= \lambda(q(4),0) = \lambda(q_1,0) = 1 \\ y(6) &= \lambda(q(5),0) = \lambda(q_1,0) = 1 \\ y(7) &= \lambda(q(6),1) = \lambda(q_1,1) = 2 \end{aligned}$$

which means that, after the first bad day, at the end of the each following time period the automaton is issuing an alert about the possible destroying of the system and finally, after the second bad day, it announce for the death of the system.

3. PREDICTING OF AN AGROBIOLOGICAL SYSTEM WITH AUTOMATON

Let us now describe an approach to use finite automaton for resolving a task in the domain of plant protection. Suppose that the breakout and the development of some plant disease depends on few (for example n) agro-meteorological, or another, *parameters* $P_1, P_2, \ldots, P_n$. Suppose also that we dispose with a corresponding *biological model*, according to which some combinations of values of the parameters have an influence on the breakout and the development of the disease—some of them strongly favor its development, some of them favor the development but not so much, another do not favor or suppress the development.

In order to register in time and prevent the development of the pathogen it is necessary to observe the values of the parameters in a sequence of *time intervals*, corresponding to the moments of a discrete time and denoted with the natural numbers—$0, 1, 2, \ldots$. The length of the time intervals has to be defined in correspondence with the biological model. Let us denote with $P_i(j)$ the value of the parameter P_i during the time moment j. If the intervals defined by the model are too long, the value of each parameter is registered many times inside the interval the value of $P_i(j)$ for j-th interval could be some aggregation of the values of the parameter in the interval—min, max, average and so on.

A typical possibility for appearing of a pathogen could be a sequence of intervals which are more or less favorable for its development. Time intervals in which the values of the parameters are favorable for the development of the disease are called *dangerous intervals*. Time intervals in which the values of the parameters are unfavorable for appearance of the pathogen are called *suppressive intervals*. The different values of parameters in suppressive as well as in dangerous intervals could have different impact on development of the pathogen. So we will choose, depending on the biological model, an integer number of *levels of development* of the disease. Let us denote them with $L_0, L_1, \ldots, L_m$. If the level of development of the pathogen becomes L_m then the plants get diseased.

Now we could classify the different values of the parameters in the dangerous intervals in m *input categories* $D_1, D_2, \ldots, D_m$ in the following way: the values from the category D_i increase the level of the development of the disease from l to $\min l + i, m$. In a similar way the different values of the parameters in the suppressive intervals could be classified in $m - 1$ input categories $S_1, S_2, \ldots, S_{m-1}$ in the following way: the values from the category $S_i, i < m$ decrease the level of the development from l to $\max l - i, 0$. It is impossible to suppress the disease if the process is achieved the level L_m. The values of the parameters that do not change the level of development of the disease are classified in a separate input category Z.

Depending on the biological model we could choose some number of high levels of development of the disease, $L_k, L_{k+1}, \ldots, L_m$, as *critical intervals*. If the system enters in such interval some reaction, for example treatment of the plants with some anti-pathogen, is absolutely necessary. It is possible also, depending on the model, to define some of the critical intervals as *fatal* when, for example, it is too late for treatment. It is possible different kinds of treatment to have different suppressive force. In such case each treatment has to be classified in one of the input categories S_j and in the first moment after the treatment some canonic values for the category S_j to be given to the system as input instead of the real values in the moment.

So, the predicting system has to input for each interval the values of the parameters, to classify them, to calculate the current level of development of the disease and to announce it for information of the responsible staff. When the process enters for first time in a critical interval the system has to issue an alert in order to inform the staff that the corresponding treatment or activity has to be applied. If as a result of treatment or unfavorable values of the parameters the process goes out of the critical intervals, the system has to inform that the immediate danger no more exists and to continue to inform for the progress of the process.

Such a predicting system could be successfully implemented with a finite automaton working in discrete moments of the time, that we denoted with the natural numbers $0, 1, 2, \ldots$. Each step of the automaton will correspond to an interval in which the parameters of the process are observed. Let $in\_code : \mathcal{P} \to \mathcal{X}$ be a coding function that maps each vector of values of the parameters:

$$\mathcal{P} = (P_1(j), P_2(j), \ldots, P_n(j)), j = 0, 1, 2, \ldots$$

in one of the input categories:

$$\mathcal{X} = \{S_{m-1}, S_{m-2}, \ldots, S_1, Z, D_1, D_2, \ldots, D_m\},$$

where $P_i(j)$ is the value of the parameters for the j-th day of the observed process. So the input alphabet of the automaton in the general case will be:

$$X = \{-m+1, -m+2 \ldots, -1, 0, 1, 2, \ldots, m\},$$

where the category S_i is represented by the input character $-i$, Z—by the input character 0, and D_j—by the input character j. It is possible that the biological model excludes the usage of some of the letters of the input alphabet.

The states of the automaton will correspond to the levels of the development of the disease. It is quite natural the state corresponding to the level L_0, i.e. missing of any danger for the plants, to be the initial state of the automaton q_0. Then the state q_l will correspond to the level l of development of the disease. The states $q_k, q_{k+1}, \ldots, q_m$ will correspond to the the critical levels of the development. For software implementation of the automaton it is more appropriate the state q_i to be replaced by the natural number i. So $Q = \{0, 1, \ldots, m\}$.

Each output characters of our automaton will reflect its current state. They will be natural numbers also $Y = \{0, 1, \ldots, m\}$ and output i will mean that the level of the development of the disease is L_i. But, for making the system user friendly, they could be replaced by more clear and descriptive texts. For example, the output 0 could be replaced by the text "The plants are in perfect condition, no danger of disease at all!"; the output $k-1$ could be replaced by the text "The level of the development of disease is very high. Any unfavorable interval could push the process to a critical phase! Be prepared for treatment!"; and so on. Let the function that translate the output character in text be out_code.

The transition function of the automaton will be presented in a table δ with $m+1$ rows, labeled with the states of the automaton and $2m$ columns labeled with the letters of the input alphabet. The value $\delta(q, x)$ in the row of labeled with q and in the column labeled with x is defined as follow:

$$\delta(q,x) = \begin{cases} \min\{q+x, m\}, & 0 \le x < m \\ \max\{q+x, 0\}, & x < 0 \\ -1, & x = m \end{cases}$$

When $\delta(q, x)$ has no sense for some q and $x \neq m$ we could remain the corresponding value $\delta(q, x)$ not defined and to mark this putting in the table $\delta(q, x) = -1$.

There is no need of a specific table for the output function λ of the automaton because $\lambda(q, x) = \delta(q, x)$ for each $q \in Q$ and for each $x \in X$ where $\delta(q, x)$ is defined and is undefined when $\delta(q, x)$ is undefined too.

When the automaton is constructed then the predicting software system executes a very simple algorithm that simulate the work of the automaton:

Algorithm:
q:=0
while $q_1 \neq -1$ **do**
    x:=in_code(input($current\ \ parameters$))
    q_1 := $\delta(q, x)$
    output(out_code(q_1))
    q := q_1
done

Using this ***Algorithm*** a universal program was created that emulates the behavior of a finite automaton and facilitates extremely creation of a software systems for plant protection. The description of any automaton could be given to the program as input. A more simple version of the program in which the description of the specific automaton is embedded was created too.

4. SUPERVISING *PHITOPHTHORA INFESTANS* ON POTATOES AND TOMATOES WITH AUTOMATON

We demonstrated the capabilities of the approach for monitoring plants healthiness with finite automata, applying it for development of a predicting system for signaling the development of mildew (*Phitophthora infestans*) on potatoes and tomatoes. Biological models that could be used for our purposes were developed for climatic conditions in Bulgaria by Popov, Mihailova, and others ([8], [6], [9], [7]). Mihailova published few different versions of conditions for the appearance and development of mildew, in which values of the favor agro-meteorological parameters vary in tight limits. As a basis for the finite automaton modeling the biological model recommended by Mihailova in [6] was chosen. It is consistent with the climatic conditions of the country and related to the biological characteristics of the disease.

Below we demonstrate how to use a finite automaton for transfer of the biological model to a language that is easy understandable by a modern computer system, with the proviso that the aim of this work is not to verify the accuracy of a particular biological model but to show the capabilities of the selected discrete mathematical tool for algorithmic simulation of such a model.

The length of the observed intervals in the used biological model is one day. As affecting agro-meteorological parameters of the disease were considered P_1—the minimum of the daily temperature, P_2—the maximum of the daily temperature and P_3—the relative daily humidity. Dangerous are classified the days with minimal temperature of the day at least 10°C, maximal temperature below 25°C and the relative humidity of the air at the same time more than 90%.

The number of levels of the development of mildew, by the biological model, was estimated to $m = 5$ days and each dangerous day leads from the level $L_i, i < m$ to the level L_{i+1}. So each such day is categorized as D_1. Day with any other combination of the tree parameters was classified as the totally suppressive S_4, i.e. does not mater on which level of the development of the disease, less than 5, happens such day the process returns to L_0. Because the supposition that predicting of the process starts at the moment when the plants are healthy, then $S_4 \equiv Z$. In addition there is an effective treatment that stops development of the disease inside a single day interval and, as it was mentioned above, we could classify the day of treatment suppressive interval too. That is why for the model of Mihaylova we have only two input categories $Z \equiv S_4$ and D_1. This lead us to an input alphabet with two letters $X = \{0, 1\}$—0 corresponds to $Z \equiv S_4$ and 1 corresponds to D_1. Finally the model consider

Following our approach we defined the following states of the automaton:

— q_0 - level L_0, the absence of danger;

— q_1 - level L_1, history with one dangerous day;

— q_2 - level L_2, history with two consecutive dangerous days;

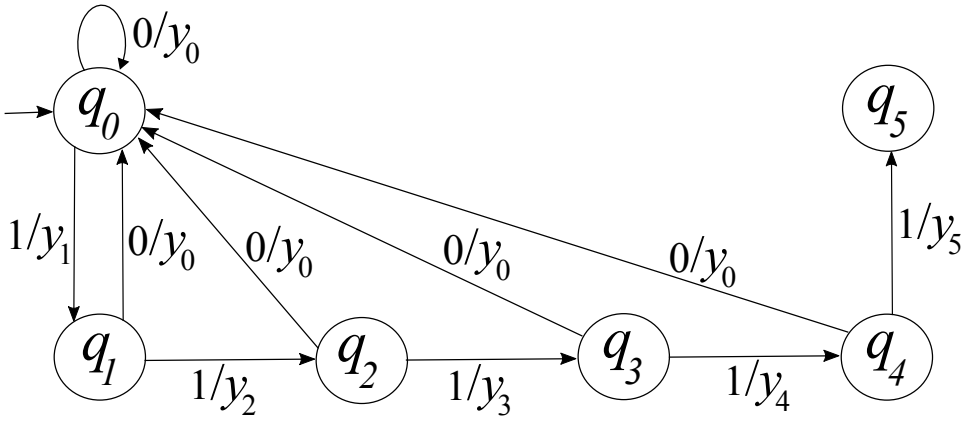

Fig. 2. Predicting finite automaton.

— q_3 - level L_3, history with three consecutive dangerous days, start of the critical period;
— q_4 - level L_4, history with four consecutive dangerous days, critical day;
— q_5 - level L_5, history with five consecutive dangerous days, irreversible destruction of the plants.

The corresponding set of messages the automaton will issue could be the following (the message $y_i, i = 0, 1, 2, 3, 4, 5$ correspond to the output letter i):

— y_0 - "There is no danger for plants.";
— y_1 - "The previous day was dangerous!";
— y_2 - "The two previous days were dangerous!";
— y_3 - "Situation becomes critical! The three previous days were dangerous. Treatment is necessary in one of the next two days unless unfavorable for the disease circumstances happen!";
— y_4 - "Situation is crucial! The previous day was critical, but there was no treatment or unfavorable for the disease circumstances. Treatment is mandatory in the current day!";
— y_5 - "Sorry, the plants are irreversibly destructed!"

The definition of the function δ of the obtained automaton is shown in Table 1 and the definition of the function λ–in Table 2.

The supervising finite automaton is shown in Fig. 2. It was enough to embed the table of its transaction function δ in the universal program that interprets a specific automaton in order to obtain the basic version of our predicting software system.

Due to the fact that, since the outbreak of the disease until its full development, they are several days, the user could predict the dynamic of the process and make the corresponding decisions. The system clearly indicated the moment, before which the treatment must be carried, and also informed the user for cancelation of the imminent danger. Thus system allowed the staff to limit treatments to the necessary minimum. This had very positive effect of biological, environmental and, last but not least, economic nature.

5. COMMENTS AND CONCLUSION

The software based on the above presented model worked in real time in the Chepintsi Computing Center for two growing seasons.

Table 1. The function δ of the obtained automaton.

δ	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_0	q_4
q_4	q_0	q_5
q_5	stop	stop

Table 2. The function δ of the obtained automaton.

λ	0	1
q_0	y_0	y_1
q_1	y_0	y_2
q_2	y_0	y_3
q_3	y_0	y_4
q_4	y_0	y_5
q_5	NO	NO

Installed in the field sensors for reporting agro-meteorological data submitted each minute the measurements to the PC they were connected and the program issued a daily prediction. These predictions were compared with the real situation occurring in the field. The climatic conditions during the period were suitable for repeated occurrences of mildew in potato and tomato planting. The system successfully predicted 87% of the occurrences of mildew and two treatments which were not necessary (because of the appearance of unfavorable conditions for the development of the disease) were cancelled due to the predictions of the system.

This experiments allows us to conclude that the proposed finite automaton modeling is suitable for predicting the breakout and development of mildew in potatoes and tomatoes.

The results of observation during the test period are not a subject of this paper. But they demonstrated that software predictions are not very correct if the measured values of the parameters are very close to the borders of the input categories. This requires the accumulation of more data for the phytopathogen reaction in such combinations of parameters values and using the mentioned above statistical methods to determine more accurate the input data categories. The more accurate definitions could be introduced in the system without reprogramming and to make it more effective.

6. REFERENCES

[1] A. Kerr and P. Keane. *Plant Pathogens and Plant Diseases*, chapter Prediction of disease outbreak, pages 299–314. Rockvale Publications, 1997.

[2] A. V. Aho, R. Sathi, and J. D. Ulman. *Compilers: Principles, Techniques, and Tools*. Addison Wesley, 1986.

[3] E. D. De Wolf, L. V. Madden, and P. E. Lipps. Risk assessment models for wheat fusarium head blight (abstract). *Phytopathology*, 90:S19, 2000.

[4] E. M. Del Ponte, C. V. Godoy, M. G. Canteri, E. M. Reis, and X. B. Yang. Models and applications for risk assessment and prediction of asian soybean rust epidemics. *Fitopatologia Brasileira*, 31:533–544, 2006.

[5] L. V. Madden. Botanical epidemiology: Some key advances and its continuing role in decease management. *European Journal of Plant Pathology*, 115:3–23, 2003.

[6] P. Mihaylova. Opportunities to use some systems to predict the appearance of mildew on potatoes (Phitophtora investa de bary) in Bulgaria (in Bulgarian). *Plant Sciences*, 11:56–61, 1965.

[7] P. Mihaylova. Relative rate of infection in mildew (in Bulgarian). *Plant Sciences*, 1:116–124, 1978.

[8] P. Popov and P. Mihaylova. *Guide for Prerdiction and Signaling Crops Pests (in Bulgarian)*. Zemizdat, Sofia, 1961.

[9] P. Popov, P. Mihaylova, G. Gospodinova, and T. Zaharieva. *Guide for Prerdiction and Signaling Crops Pests (in Bulgarian)*. Zemizdat, Sofia, 1975.

[10] S. Sankarana, A. Mishraa, R. Ehsania, and C. Davis. A review of advanced techniques for detecting plant diseases. *Computers and Electronics in Agriculture*, 72(1):1–13, 2010.

[11] P. W.Steiner. Predicting apple blossom infections by erwinia amylovora using the maryblyt model. *Acta Horticulturae*, 273:139–148, 1990.

[12] J. Yuen. *Bayesian approaches to plant disease forecasting*. Plant Health Progress (online), doi: 10.1094/PHP-2003-1113-06-RV, 2003.