Sequential pattern recognition employing recurrent fuzzy systems[☆]

Roland Kempf, Jürgen Adamy*

Institut für Automatisierungstechnik, Technische Universität Darmstadt, Landgraf-Georg-Str. 4, Darmstadt, 642 83, Germany

Abstract

Sequential pattern-recognition systems check whether data strings, e.g., time series, exhibit certain pattern primitives in a specified order. As in the case of most other pattern-recognition methods, either conventional methods or fuzzy systems may be used here. This paper presents a sequential pattern-recognition system employing recurrent fuzzy systems that is employed as a monitoring system on continuous-casting systems in the steel industry worldwide. Taking that application as a starting point, a general method for sequential pattern recognition in time series that uses recurrent fuzzy systems is described.

Keywords: Recurrent fuzzy systems; Automata; Syntactic pattern recognition; Continuous casting; Breakout prediction; Sticker detection

1. Introduction

Pattern-recognition systems are designed to check whether an object exhibits features that are similar to those of particular prototypes and thus follow a certain pattern. In addition to conventional pattern-recognition methods [9], fuzzy principles are also being successfully used in this field [7,8,28,25]. The pattern-recognition problems addressed by fuzzy pattern-recognition systems are usually static, e.g., they define and evaluate a membership function for certain patterns in the feature space involved. However, there are also syntactic pattern-recognition problems that cannot be solved using static functions. Conventional [10,9] and fuzzy [29,13] methods for solving these problems exist. One type of syntactic pattern-recognition problems are sequential pattern-recognition problems

☆ Research partially supported by the Deutsche Forschungsgesellschaft under grant AD 174/2-1.
* Corresponding author. Tel.: +49-6151-16-3442; fax: +49-6151-16-2507.
E-mail address: adamy@rtr.tu-darmstadt.de (J. Adamy).

[22,10] similar to string-matching problems [9] that test whether a specified series of pattern primitives occurs in an object, e.g., a time series. If one would like to use fuzzy principles for solving such pattern-recognition problems, one may employ fuzzy systems, such as recurrent fuzzy systems [1,2,5], that have an inherent dynamic character.

In this paper, we present a general configuration for sequential pattern-recognition systems employing recurrent fuzzy systems that may be approximately described in terms of finite automata and add to the theory of recurrent fuzzy systems that was initiated in [4,15]. The basic concepts of recurrent fuzzy systems were independently and contemporaneously developed and published in [1,2,11]. In [1,2], the motivation was such systems' similarity to automata.[1] In [11] the motivation was their relationship to neural networks. Refer to [33,38,26] for further literature on recurrent fuzzy systems and their relationship to neural networks. We will start off by discussing some basic properties of sequential pattern recognition and recurrent fuzzy systems in order to develop the general configuration for the desired fuzzy sequential pattern-recognition system. We will present a real-world application of such a system that is used to monitor continuous-casting processes in the steel industry [1,2]. Finally, we will analyze its configuration and extend it in order to be able to present an approximate description of the resultant system in terms of a finite automaton.

2. Fundamentals of sequential pattern recognition

Sequential pattern-recognition systems search sets of data, e.g., sets of data defining curves or character strings, for a specified series of pattern primitives [22,10]. The curves or character strings involved are frequently generated online and the data involved arrives sequentially, i.e., in temporal order. Finite, deterministic, automata are frequently used for the sequential recognition of such patterns [22,10]. The following example will illustrate how such an automaton recognizes patterns in a character string.

Assume that this paper is to be searched for occurrences of the word "curve." This paper then represents the character string to be searched and the word "curve," consisting of the pattern primitives "c," "u," "r," "v," and "e," represents the pattern sought. The input data to the sequential pattern-recognition system are the individual characters appearing in this paper in the order in which they appear herein. The sequential pattern-recognition system compares them in turn to one of the letters "c," "u," "r," "v," or "e." Its state variable, x, tells it to which letter of the word "curve" it should compare the characters contained in the text entering its input. If the letters of the word "curve" arrive one by one in the correct order, each letter will cause the value of x to be incremented by one, commencing with $x = 1$, until the value $x = 6$ is reached, after a series of five successive

[1] At the request of the editors, we would like to explain the change in the designation of the systems treated in this article from the "fuzzy automata" used in 1995 in [1,2] to the "recurrent fuzzy systems" used in [4,15] and this article: They were initially called "fuzzy automata" in order to express their relationship to automata. Since whether these fuzzy systems behave like automata or more general dynamic systems depends on their design, in 1997, we called them "dynamic fuzzy systems" [5]. Unfortunately, this name is also used for another type of dynamic fuzzy systems [16,31]. Thus, searching for a suitable name, we renamed them "recurrent fuzzy systems" in 2000. After a thorough literature search using this designation, we found an article by Gorrini and Bersini [11], where they had used the designation "recurrent fuzzy system" for the same system structure used in [1,2] as early as 1994. In our opinion, this latter designation is the best choice.

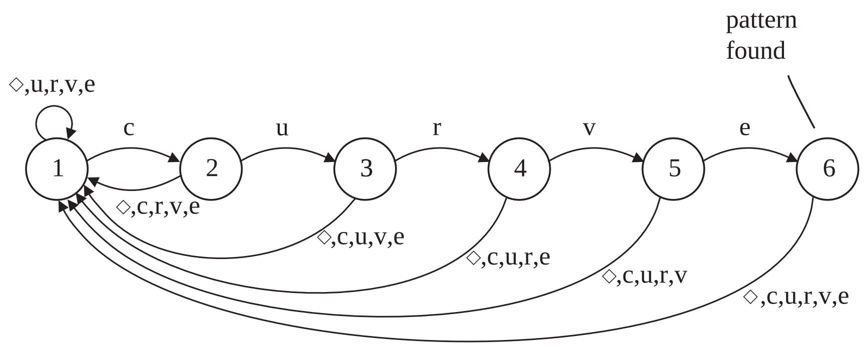

Fig. 1. Transition graph for a sequential pattern-recognition system based on a finite automaton.

incrementations. Incidences of x reaching the value $x = 6$ will indicate that all five letters of the word "curve" have been detected in sequence and in the correct order. If a wrong letter should appear during the search, the value of x will be reset to $x = 1$ and the system will start again to search for a new appearance of the word "curve" in the text.

The behavior of this sequential pattern-recognition system may be readily read off its state graph, which is shown in Fig. 1, where the values of its state variable, x, are represented by nodes. The arrows indicate transitions in the value of its state variable, x, caused by arrival of the respective input parameters noted above the arrows at its input, where the "◇"-symbol collectively represents all letters and other typeset characters other than "c," "u," "r," "v," or "e." It is useful to distinguish the following categories of rules causing the different transitions: "Transition rules" are rules that lead to an increased value of the state variable x. "Reset rules" interrupt the recognition process in case of a wrong letter and reset the automaton. Finally, the "end rule" resets the automaton to its initial state, $x = 1$, if the recognition process was finished successfully.

It should be noted that this setup might miss certain occurrences of the word sought in a text if the word involved is part of another word. Assume, for example, that the fictitious character string "curcurve" is to be searched for occurrence of the word "curve": Once the pattern-recognition process arrives at the second "c" in "curcurve" it will abort the first recognition process, since the letter "v" was expected. However, the letter "c" will not be recognized as part of a new pattern. How such problems may be dealt with will be found in [9].

Automata may also be used to search for other types of character strings, such as strings of binary code [35] or strings occurring in regular, formal languages [10,9], employing similar procedures. Patterns occurring in curves may also be recognized using the same approach, in which case, the pattern sought is described by a series of pattern primitives. A typical example hereof is recognizing patterns in electrocardiograms (ECG) [10,17]. The pattern primitives are labeled with characters, as shown in the example appearing in Fig. 2. The pattern sought may thus be represented by a character string or a word, in the case of this particular example, the word "c–u–r–v–e." Making suitable choices of the other pattern primitives and their designations by characters will also allow representing the curve to be scanned by a character string. Searching that character string for the word "c–u–r–v–e" will then be equivalent to searching the curve for occurrences of the pattern shown in Fig. 2. We shall now outline a procedure for describing the curve's pattern primitives in

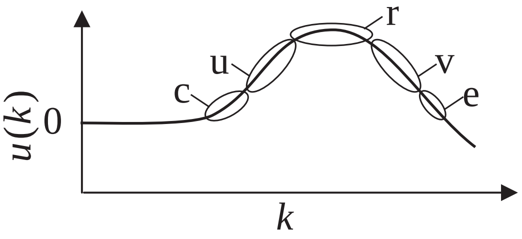

Fig. 2. Example of a pattern occurring in a time series, $u(k)$, subdivided into several pattern primitives labeled with the letters “c,” “u,” “r,” “v,” and “e.”

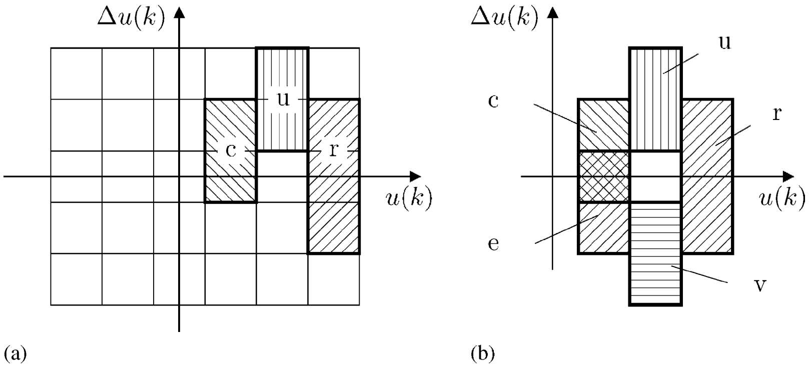

Fig. 3. Input space of a sequential pattern-recognition system. The labeled sections represent the features of pattern primitives.

a manner that will allow assigning characters to them. In the case of the example stated above, this procedure corresponds to extracting the feature primitives, labeled “c,” “u,” “r,” “v,” and “e,” of the pattern to be recognized from the input data.

The starting point for describing pattern primitives is the database available for pattern-recognition purposes. Curves are usually defined by a time series, $u(k)$. The pattern primitives thus must be described by the numerical values $u(k)$ and other values derived therefrom, e.g., the differences $\Delta u(k) = u(k) - u(k-1)$. One means for defining pattern primitives is assigning them to certain ranges of the quantities $u(k)$ and $\Delta u(k)$, which necessitates a partitioning of the input space, U, which is spanned by the quantities $u(k)$ and $\Delta u(k)$, as may be seen from Fig. 3(a). Suitable ranges for all those pattern primitives sought in the example mentioned above have been indicated in Fig. 3(b). The pattern primitive “c” may then be represented by, e.g., “a slight increase in u for slightly increased values of u, i.e., by the vectors $\mathbf{u}(k) = (u(k), \Delta u(k))$, which lie within the set designated in Fig. 3. In this case as well, once the characters have been generated from the input signals using the preprocessing procedure described above, the sequential pattern-recognition system will operate in the same manner as a finite automaton. In practice, however, the signals involved will normally be scanned, noisy, signals. How these signal-processing problems may be overcome will be described below.

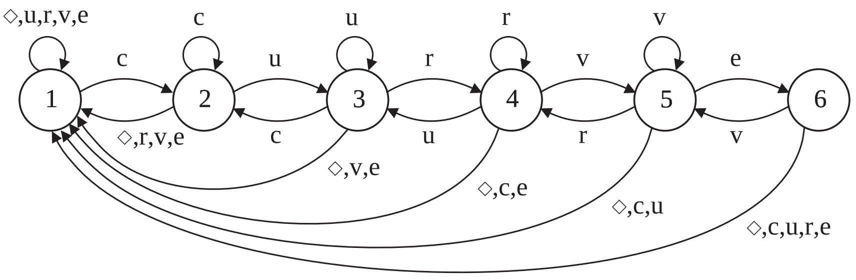

Fig. 4. Transition graph for a sequential pattern-recognition system based on a finite automaton that incorporates hold and backstep rules.

In the case of scanned signals, the values of input parameters will normally be correlated to the same character over several scanning increments if the signal amplitude changes only slightly over a scanning increment. Additional "hold rules" are introduced in order to make the pattern-recognition system immune to repetitions of the same pattern primitive. If the pattern-recognition system considered above has, e.g., already detected the letter "c" and is in the state $x = 2$, it will then proceed to search for the letter "u." If it detects the letter "c" again, it will remain in the state $x = 2$, rather than abort the search.

In the case of noisy signals, problems occur at transitions between ranges assigned to different characters. If a noisy signal crosses, e.g., the boundary between the ranges for the letters "c" and "u" during the process of pattern recognition, it is highly likely that boundary of the range for the letter "c" will be briefly recrossed during one of the next steps before it returns to the range for the letter "u" due to the noise. One means for preventing abortion of the pattern-recognition process is briefly decrementing the value of the state variable, $x = 3$, to $x = 2$, and then continuing the pattern-recognition process. Such "backstep rules" make pattern recognition less sensitive to noise.

These additional types of rules, i.e., hold rules and backstep rules, may be simply incorporated into the automaton involved. Fig. 4 depicts the modified state graph for this case. However, the new rules impose further conditions on the approach to sequential pattern recognition. Since for every conceivable situation it must be clear which type of rule applies, the ranges in the input space, U, that lead to the character sets for the hold, backstep, and transition rules may not intersect one another.

Although the approach to sequential pattern recognition described above is based on an example, it may be generalized as follows: The sequential pattern-recognition system is a finite automaton that recognizes a pattern composed of $j_{\max} - 1$ pattern primitives. In order to be able to accomplish that task, it needs a state variable, x, that can take on the values L_1^x to $L_{j_{\max}}^x$, which, in the case of the example considered above, are the values 1 to $j_{\max}$. For every $j \in \{2, 3, \ldots, j_{\max}\}$, the $(j-1)$th pattern primitive of the total of $j_{\max} - 1$ pattern primitives is described by a set, $B(L_j^x)$, in the input space, U. In addition, the set $B(L_1^x) = U \setminus B(L_2^x)$ describes the input values that, in the initialization state, $x = L_1^x$, do not indicate the commencement of the series of pattern primitives sought in the signal. Finally, the set $A(L_j^x) = U \setminus (B(L_{j-1}^x) \cup B(L_j^x) \cup B(L_{j+1}^x))$ describes the set of inputs that will trigger a reset to the initialization state during a search for the $(j-1)$th pattern primitive. This notation allows compactly summarizing the individual rules in the form shown in Table 1. Only one of the rules will apply to any given situation, and that rule will thus uniquely specify the state

Table 1
Rule base for a conventional sequential pattern-recognition system

Rule type	$x(k)$	$\mathbf{u}(k)$	$x(k+1)$
Transition rule	$L_j^x, j<j_{\max}$	$B(L_{j+1}^x)$	L_{j+1}^x
Hold rule	$L_j^x, j<j_{\max}$	$B(L_j^x)$	L_j^x
Backstep rule	$L_j^x, j>1$	$B(L_{j-1}^x)$	L_{j-1}^x
Reset rule	L_j^x	$A(L_j^x)$	L_1^x
End rule	$L_{j_{\max}}^x$	$B(L_{j_{\max}}^x)$	L_1^x

transition the automaton is to undergo. The general structure and the application of the rules to be employed in designing a sequential pattern-recognition system have thus been established.

One of the major tasks remaining in designing a sequential pattern-recognition system is extracting the pattern primitives from the input signal, $u(t)$, i.e., extracting the characters involved. In the case of the example considered above, the pattern primitive “c” is verbally described as “a slight increase in u for slightly larger values of u.” The pattern primitive “u” may be described as “a slight to large, positive, slope for moderate, positive, values of u.” The other pattern primitives may be described in similar fashion. Instead of modeling these verbal descriptions employing conventional sets, they may naturally be represented by fuzzy sets. In this case, the fuzzification of the input variables will systematically yield the feature extraction. The results of those feature extractions will be fuzzy sets, $B(L_j^x)$, in the input space, U. An initial approach to processing the input signal might involve interpreting the rules of Table 1 as fuzzy rules, rather than as switching rules for an automaton, and treating the value of the state variable, x, as a fuzzy variable. The value of the state variable might then also fall anywhere between the values, L_j^x, contained in the former finite set of values and thus indicate both that a certain number of pattern primitives has been detected and to which extent there are signs of a subsequent feature primitive. In the case of slowly varying input values, there will thus be a quasicontinuous transition in the value of the state variable, x, provided that no reset rules that would reset the state involved to the initialization state apply.

Owing to this extension of the finite state space to a continuum, i.e., $x\in\mathbb{R}$, instead of $x\in\mathbb{N}$, a sequential pattern-recognition system configured in this manner will no longer constitute a finite automaton, but a fuzzy system with output feedback, i.e., a recurrent fuzzy system. A formal definition and investigation of recurrent fuzzy systems is given in [4,15]. In the following section, we shall treat several fundamental aspects of recurrent fuzzy systems that are essential to understanding the approach to sequential pattern recognition employing recurrent fuzzy systems to be described further below.

3. Fundamentals of recurrent fuzzy systems

The configuration of recurrent fuzzy systems is similar to that of finite, multivalued automata [4]. Both share the same block schematic[2] (cf. Fig. 5) and describe a transition of a real-valued state vector, $\mathbf{x}(k)$, to a new state vector, $\mathbf{x}(k+1)$, at time increment $k+1$ depending upon an input vector,

[2] In [4], we introduced a more general configuration that included an output function $\mathbf{y}=\mathbf{g}(\mathbf{x},\mathbf{u})$.

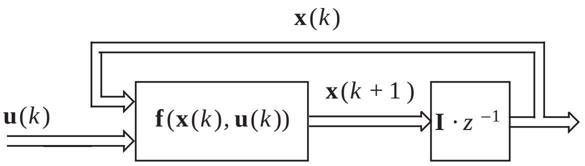
$\mathbf{x}(k)$
$\mathbf{u}(k)$
$\mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))$
$\mathbf{x}(k+1)$
$\mathbf{I} \cdot z^{-1}$

Fig. 5. Block schematic of an automaton and a recurrent fuzzy system. **I** denotes the unit matrix.

$\mathbf{u}(k)$, using the equation $\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))$. However, the transition function, $\mathbf{f}$, resembles a static fuzzy function, rather than a function being assembled from logical switching functions, as in the case of finite automata. In some cases, recurrent fuzzy systems behave similarly to finite automata, and in other cases they behave rather differently, as described in [4]. A fuzzy transition function may be obtained in the following manner:

A fuzzy transition function provides a mathematical model for a set of fuzzy rules describing the transition of the state vector, $\mathbf{x}$. In the rule base containing these rules, each state variable, x_i, is characterized by a set of linguistic values, $L_{j_i}^{x_i}$, and each input variable, u_p, is characterized by a set of linguistic values, $L_{q_p}^{u_p}$. The rule base should be complete, free of contradictions, and contain exclusively rules that have the following form:

$$\begin{aligned}&\text{if } x_1(k) = L_{j_1}^{x_1} \text{ and} \ldots \text{and } x_n(k) = L_{j_n}^{x_n}\\ &\text{and } u_1(k) = L_{q_1}^{u_1} \text{ and} \ldots \text{and } u_m(k) = L_{q_m}^{u_m},\\ &\text{then } x_1(k+1) = L_{w_1}^{x_1} \text{ and} \ldots \text{and } x_n(k+1) = L_{w_n}^{x_n}.\end{aligned} \tag{1}$$

A combination of linguistic values, $L_{j_i}^{x_i}$, for every index, i, e.g., $\mathbf{L}_j^x = (L_{j_1}^{x_1}, L_{j_2}^{x_2}, L_{j_3}^{x_3})$, is also termed a "linguistic vector," $\mathbf{L}_j^x$. The same remark applies to the linguistic values $L_{q_p}^{u_p}$ involved. The rule base thus defines a mapping of the linguistic vectors $\mathbf{L}_j^x$ and $\mathbf{L}_q^u$ onto linguistic vectors $\mathbf{L}_w^x = \mathbf{L}_{w(j,q)}^x$, and may be visualized using a state graph of a "linguistic automaton" [4] that utilizes the linguistic vectors $\mathbf{L}_j^x$ as discrete states.

In order to arrive at the necessary mathematical model, we design a static fuzzy function, $\mathbf{f}$, that includes the operations of fuzzification, inference, and defuzzification, defining the relation

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)). \tag{2}$$

During fuzzification, every linguistic value, $L_{j_i}^{x_i}$, of the state variables, x_i, is associated with a convex membership function, $\mu_{j_i}^{x_i}(x)$, where the values of these membership functions add up to unity at every point, x_i, i.e., $\sum_{j_i} \mu_{j_i}^{x_i}(x_i) = 1$. A so-called "core position," $s_{j_i}^{x_i}$, is selected from every core for which the membership function, $\mu_{j_i}^{x_i}(x_i)$, takes on its maximum value of 1. This core position, $s_{j_i}^{x_i}$, also serves as the singleton position during defuzzification for the linguistic value $L_{j_i}^{x_i}$. It thus follows that $L_{j_i}^{x_i}$ is associated with a pair of membership functions, the membership function, $\mu_{j_i}^{x_i}$, appearing in the fuzzification part for obtaining the fuzzified values of the state variables, $x_i(k)$, fed back to the input of the fuzzy system and the singleton, $s_{j_i}^{x_i}$, appearing in the defuzzification part for obtaining values of the output parameters, $x_i(k+1)$, of the fuzzy system. A combination of core positions, $s_{j_i}^{x_i}$, for every index, i, is termed a "core position vector," $\mathbf{s}_j^x$.

A similar rule applies to each of the input parameters, u_p, involved and their linguistic values, $L_{q_p}^{u_p}$. We thus obtain a set of membership functions, $\mu_{q_p}^{u_p}(u_p)$, and associated core positions, $s_{q_p}^{u_p}$,

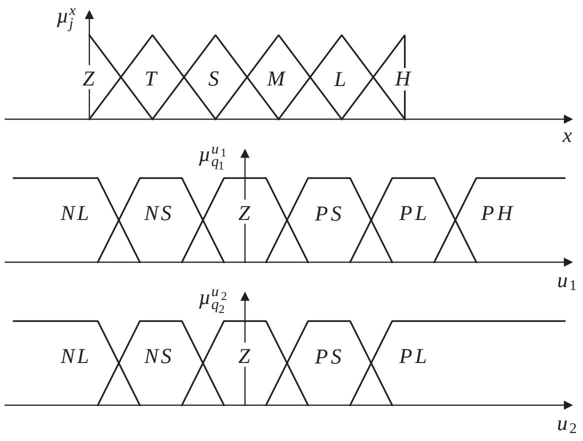

Fig. 6. Typical membership functions for the state variable, x, and input variables, u_p, of a recurrent fuzzy system.

where these membership functions, $\mu_{q_p}^{u_p}(u_p)$, take on their maximum values of 1, along with a set of core position vectors, $\mathbf{s}_q^u$, for every index vector $\mathbf{q}$. Fig. 6 depicts an example of such membership functions, $\mu_{j_i}^{x_i}(x_i)$ and $\mu_{q_p}^{u_p}(u_p)$.

If we apply the PROD-SUM operators in conjunction with the inference operation and use those core position vectors, $\mathbf{s}_{w(j,q)}^x$, that correspond to the linguistic vectors, $\mathbf{L}_{w(j,q)}^x$, appearing in the conclusions as singletons in the CoS-defuzzification method [4,37], we obtain the following compact, analytical, form for $\mathbf{f}$:

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \sum_{\mathbf{j},\mathbf{q}} \mathbf{s}_{w(j,q)}^x \cdot \prod_i \mu_{j_i}^{x_i}(x_i) \cdot \prod_p \mu_{q_p}^{u_p}(u_p). \tag{3}$$

If the membership functions are continuous, $\mathbf{f}(\mathbf{x},\mathbf{u})$ will also be continuous and the resultant recurrent fuzzy system is said to be continuous. A more detailed description and analysis of recurrent fuzzy systems is given in [4].

Now that we have briefly characterized recurrent fuzzy systems, the next section will show how we may obtain a sequential pattern-recognition system employing a recurrent fuzzy system.

4. Sequential pattern recognition employing recurrent fuzzy systems

The fuzzy sequential pattern-recognition system presented in this section, which was introduced in a first version in [1–3], is basically a recurrent fuzzy system that utilizes the rules for a sequential pattern-recognition system listed in Table 1 in Section 2. In configuring this system, we shall follow the procedures for defining a recurrent fuzzy system appearing in the immediately preceding section.

We start off by choosing a number of linguistic variables, L_j^x, for the state variable, x, used as the counter variable in the sequential pattern-recognition system. Useful linguistic terms are, e.g., $L_1^x =$ “first,” $L_2^x =$ “second,” …, and $L_{j_{\max}}^x =$ “last,” or $L_1^x =$ “zero” (Z), $L_2^x =$ “tiny” (T), …, and $L_{j_{\max}}^x =$ “huge” (H).

The linguistic terms used for describing the features, u_p, to be involved may be strongly dependent on the semantic background, e.g., values of $L_1^{u_1}$ = "cool," $L_2^{u_1}$ = "warm," $L_3^{u_1}$ = "hot," and $L_4^{u_1}$ = "extremely hot," for a variable, u_1, specifying a temperature. Alternatively, generic linguistic values, such as "negative huge" (NH), "negative large" (NL), "negative small" (NS), "zero" (Z), "positive small" (PS), "positive large" (PL), and "positive huge" (PH), may also be used.

In order to obtain the rule base for the fuzzy sequential pattern-recognition system, we start off with the rules for the conventional sequential pattern-recognition system listed in Table 1. However, according to Eq. (1), the rule base for a recurrent fuzzy system must consist of rules that employ the linguistic values, $L_{q_p}^{u_p}$, of the linguistic input variables involved, rather than pattern primitives, $B(L_j^x)$, in their premises. We thus express the pattern primitives, $B(L_j^x)$, in terms of the linguistic values, $L_{q_p}^{u_p}$, of the linguistic input variables, u_p. We obtain a first expression for them by considering a verbal description of the pattern primitive, e.g.,

"The third-feature primitive, $B(L_4^x)$, is characterized by a positive-large ($=L_5^{u_1}$) first-feature variable, u_1, and a negative-small ($=L_2^{u_2}$), zero ($=L_3^{u_2}$), or positive-small ($=L_4^{u_2}$) second-feature variable, u_2."

This verbal description may then be transformed into a logical combination of the linguistic values of the linguistic input variables, e.g.,

$$B(L_4^x) = L_5^{u_1} \wedge (L_2^{u_2} \vee L_3^{u_2} \vee L_4^{u_2}). \tag{4}$$

These logical expressions are subsequently rewritten in their disjunctive normal form, e.g.,

$$B(L_4^x) = (L_5^{u_1} \wedge L_2^{u_2}) \vee (L_5^{u_1} \wedge L_3^{u_2}) \vee (L_5^{u_1} \wedge L_4^{u_2}). \tag{5}$$

In a final step, we insert the expression obtained for the pattern primitives, $B(L_j^x)$, into the rules for the sequential pattern-recognition system and split the result into a number of fuzzy rules. E.g., for the expression $B(L_4^x) = (L_5^{u_1} \wedge L_2^{u_2}) \vee (L_5^{u_1} \wedge L_3^{u_2}) \vee (L_5^{u_1} \wedge L_4^{u_2})$ and the transition rule "If $x(k) = L_3^x$ and $\mathbf{u}(k) = B(L_4^x)$, then $x(k+1) = L_4^x$," we obtain the following fuzzy rules:

If $x(k) = L_3^x$ and $u_1(k) = L_5^{u_1}$ and $u_2(k) = L_2^{u_2}$, then $x(k+1) = L_4^x$.
If $x(k) = L_3^x$ and $u_1(k) = L_5^{u_1}$ and $u_2(k) = L_3^{u_2}$, then $x(k+1) = L_4^x$.
If $x(k) = L_3^x$ and $u_1(k) = L_5^{u_1}$ and $u_2(k) = L_4^{u_2}$, then $x(k+1) = L_4^x$.

Since the rule base regards each rule as correlated to the others by an or-operator, the or-operators in the disjunctive normal form describing the pattern primitives, $B(L_j^x)$, are implicitly included in the rule base. This process yields a complete rule base that is free of contradictions, as demanded for the case of a recurrent fuzzy system.

As in the case of general recurrent fuzzy systems [4,15], the rule base may be illustrated by drawing a state graph whose nodes represent the linguistic states of the fuzzy sequential pattern-recognition system. The transition pointers are labeled with the transition conditions, i.e., the linguistic input values that will cause such a transition from one linguistic state, $x(k)$, to the next state, $x(k+1)$. An example of such a state graph having linguistic state variables Z, T, S, M, L, and H is shown in Fig. 7, where the wavy nodes represent the fuzzy nature of the rules. Since the fuzzy sequential pattern-recognition system uses the same rules as the conventional pattern-recognition system, it comes as no surprise that their respective state graphs shown in Figs. 7 and 4 are configured in the same way.

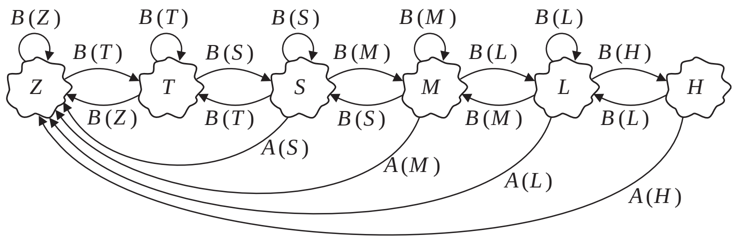

Fig. 7. Transition graph for a sequential pattern-recognition system based on a recurrent fuzzy system.

In order to define the transition function, f, we need only choose membership functions and core positions for all linguistic values. In the case of the counter, x, the membership functions, μ_j^x, are chosen to be equidistant triangular functions with respective core positions at their peaks. The linguistic values for the feature variables, u_i, are modeled by trapezoidal membership functions. An example thereof is shown in Fig. 6.

At this point, all parts of the fuzzy sequential pattern-recognition system will have been established and the relation sought, $x(k+1) = f(x(k), \mathbf{u}(k))$, will be uniquely defined by Eq. (3).

As an illustration of a sample application of this basic configuration of a fuzzy sequential pattern-recognition system, the next section will describe use of such a system for monitoring a process employed in the steel industry.

5. An industrial application to steel making

We will now illustrate the configuration of a fuzzy sequential pattern-recognition system, based on the example of such a system that has been successfully used for breakout prediction in continuous-casting operations at steel plants in South Africa, India, and Germany for several years [1,2], cf. also [30]. The system employed is a commercially available product known as a “breakout-prediction system” marketed under the trade name “B.O.P.S.” and constitutes part of the plants’ process-control system.

In the continuous-casting process, molten steel is transported to the casting lines in a ladle and poured into a tundish that distributes the molten steel to several casting lines. The molten steel runs down into an externally cooled casting mould. The level of molten steel in the casting mould is accurately maintained by a controller [18,23,24] in order to prevent level variations that might adversely affect steel quality and cause breakouts. As molten steel passes through the cooled casting mould, it starts to solidify, commencing at its lateral surfaces, which are in contact with the casting mould, and proceeding inward, forming a shell. This shell prevents the molten steel remaining in the interior of the strand from escaping as the solidified shell is continuously withdrawn from the casting mould. Once the withdrawn steel strand has fully solidified, it is cut up into slabs. Fig. 8 schematically depicts the process involved.

Problems related to formation of the steel strand’s shell in the casting mould may arise. Defects in the growth of its shell result in weak spots that can cause serious problems, since the strand may break open after it has exited the casting mould. Molten steel can escape from these breakouts and

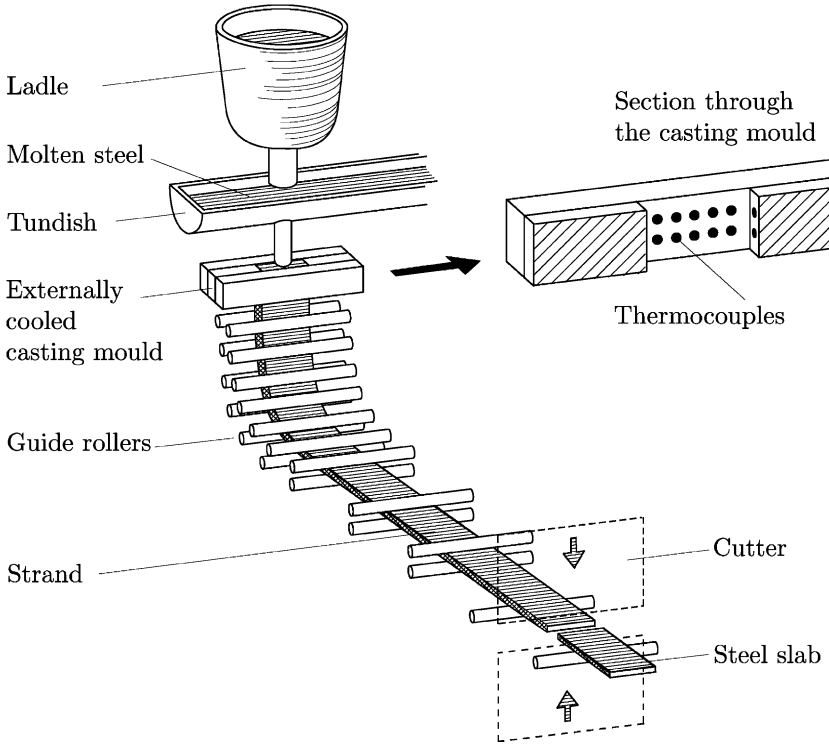

Fig. 8. A schematized depiction of the continuous-casting process.

cause serious damage to the plant's equipment, which inevitably leads to downtimes and expensive repairs.

In addition to other factors [36], the most common types of growth defects are caused by the so-called "stickers" [36,21,20] due to localized shortages of the casting powder used to reduce friction between the strand and the casting mould. The increased friction at such locations causes the strand to adhere to the inner wall of the casting mould more strongly than elsewhere, which reduces its rate of transport at those locations. The resultant stress induced in the strand's shell causes its shell to rupture. Molten steel then comes into contact with the inner wall of the casting mould, raising its temperature at that location. The rise in temperature at that location will be followed by a drop in temperature to below the normal level once material from upstream reaches that location, since it will have acquired a thicker shell due to the reduction in rate of transport and resultant longer cooling period. The changes in temperature involved may be measured using thermocouples embedded in the inner walls of the casting mould. A typical temperature curve for a sticker as a function of time is depicted in Fig. 9(a). Once a sticker has been detected, the transport rate may be temporarily reduced in order to extend the cooling period for the thin shell while it remains in the mould and thereby prevent incidence of a breakout [27]. A monitoring system is thus needed in order to allow online searches for characteristic patterns in temperature curves originating from stickers. Various methods for handling this task are known, some of which directly use measured temperatures or

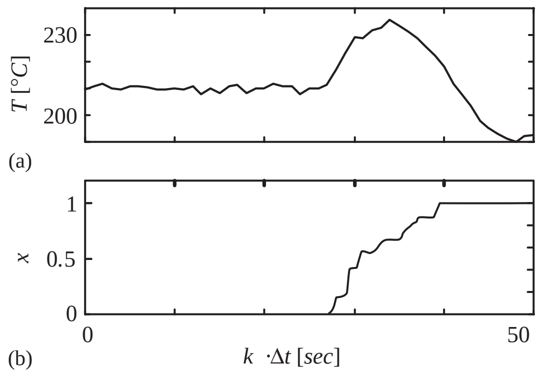

Fig. 9. Plots of the temperature curve (a) and the subjective probability of a breakout (b).

their temporal rates of change [6,36,19], while others use pattern-recognition techniques that employ neural networks [14,34,12] or other methods [32].

The system described here [1,2] employs recurrent fuzzy systems for sequential recognition of patterns correlated to stickers.[3] It is particularly noted for the low expenditures of time and effort required for its implementation and its highly reliable detection of stickers. The instantaneous temperature of the casting mould, $T(k)$, and temporal changes in its temperature, $\Delta T(k) = T(k) - T(k-1)$, are employed for characterizing its temperature as a function of time. Its instantaneous temperature, $T(k)$, is employed as an initial input parameter, u_1, to the sequential pattern-recognition system, and is described by the linguistic values "negative large" ($L_1^{u_1} = NL$), "negative small" ($L_2^{u_1} = NS$), "zero" ($L_3^{u_1} = Z$), "positive small" ($L_4^{u_1} = PS$), "positive large" ($L_5^{u_1} = PL$), and "positive huge" ($L_6^{u_1} = PH$). Changes in its temperature, $\Delta T(k)$, are employed as a second input parameter, u_2, and are described by the linguistic values "negative large" ($L_1^{u_2} = NL$) through "positive large" ($L_5^{u_2} = PL$).

The state variable of the sequential pattern-recognition system is defined in terms of a counter variable, x, which represents a subjective, i.e., nonmathematical, estimate of the probability of a breakout, and is described by the six linguistic values "zero" ($L_1^x = Z$), "tiny" ($L_2^x = T$), "small" ($L_3^x = S$), "moderate" ($L_4^x = M$), "large" ($L_5^x = L$), and "huge" ($L_6^x = H$).

The first feature primitive that should trigger a transition in the counter variable, x, from Z to T is characterized by a slight increase in temperature at a slightly higher temperature value. The range $B(T)$ is thus described by a fuzzy set "$(T = PS) \wedge (\Delta T = (Z \vee PS))$." The range, $B(S)$, of the second feature primitive, S, is described by the fuzzy set "$(T = PL) \wedge (\Delta T = (PS \vee PL))$."

The verbal descriptions of the other feature primitives that will cause the verbal stages M, L, and H to be reached, result in the following fuzzy sets for the remaining ranges, $B(M)$, $B(L)$, and $B(H)$, as well as of $B(Z)$:

$$B(Z) = U \backslash B(T),$$
$$B(T) = (u_1 = PS) \wedge (u_2 = (Z \vee PS)),$$
$$B(S) = (u_1 = PL) \wedge (u_2 = (PS \vee PL)),$$
$$B(M) = (u_1 = PH) \wedge (u_2 = (NS \vee Z \vee PS)),$$

[3] The commercially available system [1,2] also detects "cracks" [35], which are another cause of breakouts.

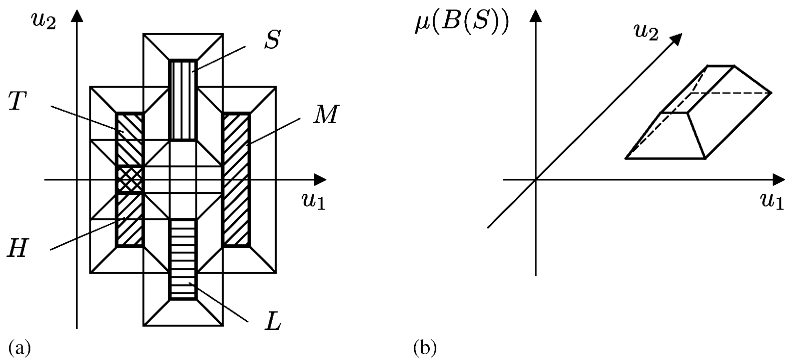

Fig. 10. (a) Top view of the membership functions for the fuzzy sets, $B(L_j^x)$, characterizing feature primitives in the feature space U. (b) The membership function for the fuzzy set $B(L_3^x) = B(S)$.

$$B(L) = (u_1 = PL) \wedge (u_2 = (NL \vee NS)),$$
$$B(H) = (u_1 = PS) \wedge (u_2 = (NS \vee Z)).$$

The entire rule base for the sequential pattern-recognition system involved may be assembled using the foregoing definitions of the ranges $B(L_j^x)$. The rules that result will later be recast in the form of Eq. (1) in order to bring them into conformity with the representation used by recurrent fuzzy systems. The following example of a transition rule will be considered:

If $x(k) = Z$ and $u(k) = B(T)$ then $x(k+1) = T$.

The first step involves recasting the descriptions of the ranges in their disjunctive normal form. For example, for the range $B(T)$, this recasting yields the expression $B(T) =$ "$((u_1 = PS) \wedge (u_2 = Z)) \vee ((u_1 = PS) \wedge (u_2 = PS))$." Every rule of the sequential pattern-recognition system will then be split into a number of rules for the recurrent fuzzy system. The following pair of rules are obtained for the transition rule under consideration:

If $x(k) = Z$ and $u_1(k) = PS$ and $u_2(k) = Z$ then $x(k+1) = T$.
If $x(k) = Z$ and $u_1(k) = PS$ and $u_2(k) = PS$ then $x(k+1) = T$.

The or-operation appearing in the description of the range $B(T)$ is carried out in conjunction with the fuzzy system's accumulation operation, which correlates every one of the individual rules by an or-operator. Applying a similar procedure to the ranges $B(L_j^x)$ and $A(L_j^x)$, we ultimately obtain a complete rule base in the form of Eq. (1) that will be free of contradictions.

The next step is modeling the linguistic values L_j^x, $L_{q_1}^{u_1}$ and $L_{q_2}^{u_2}$, which involves selecting the membership functions μ_j^x, $\mu_{q_1}^{u_1}$, and $\mu_{q_2}^{u_2}$, typical examples of which are shown in Fig. 6. Once these have been chosen, the ranges in input space will also be known in the form of fuzzy sets. Fig. 10(a) depicts a top view of the membership functions for the ranges, $B(L_j^x)$, and Fig. 10(b) depicts the membership function for the range $B(L_3^x) = B(S)$.

Finally, Fig. 9(a) depicts a temperature curve for a "sticker" that is to be recognized by the sequential pattern-recognition system. In Fig. 9(b), the evolution of the state, x, of the pattern-recognition system is shown. It passes all core positions, s_j^x, corresponding to the linguistic values "zero" ($L_1^x = Z$) through "huge" ($L_6^x = H$) and the pattern will thus be recognized. Furthermore, the quasicontinuous curve for the degree of recognition, x, indicates that the state of the sequential pattern-recognition system might well fall between the defined verbal stages, Z, T, S, M, L, and H. In such intermediate ranges, although the next pattern primitive will, in some sense, be detected, it will not have been unambiguously detected by that time.

The recurrent fuzzy system employed represents a nonlinear, time-discrete, system. More recent theoretical results [4,15] allow investigating its dynamic behavior in greater detail. Of particular interest is whether this particular recurrent fuzzy system's behavior is somewhat similar to that of the linguistic automaton whose transitions are determined by the rule base that has been defined.

6. Properties of the fuzzy sequential pattern-recognition system

The operation of sequential pattern-recognition systems involving conventional automata, like those described in Section 2, may be readily understood by following the respective associated rule, which also applies to the case of a recurrent fuzzy system, provided that only a single rule is active at any given time. Its behavior will then be equivalent to that of the automaton [4]. However, several rules will frequently be simultaneously active in the recurrent fuzzy system. Its behavior will, under some circumstances, then be largely dependent upon the weightings of the active rules. If its behavior may, nevertheless, be approximately described by a single rule from the rule base, then the recurrent fuzzy system is said to be "automaton-like." A formal definition and a discussion of the properties attributable to this similarity to an automaton are given in [4].

For the convenience of readers, the meaning of the term "automaton-like" will be briefly reviewed below, for which we shall need the following definition of correlations between core-position vectors and other vectors: A vector, e.g., $\mathbf{x}$, is said to be correlated to a core-position vector, e.g., $\mathbf{s}_j^x$, if the components, $s_{j_i}^{x_i}$, of the core-position vector may be obtained either by rounding the individual components, x_i, of the vector $\mathbf{x}$ up to the nearest higher core position or rounding them off to the nearest lower core position. The respective components, x_i and $s_{j_i}^{x_i}$, of the pair of correlated vectors, $\mathbf{x}$ and $\mathbf{s}_j^x$, thus lie close to one another and the core-position vector, $\mathbf{s}_j^x$, correlated to the vector $\mathbf{x}$ approximately describes the vector $\mathbf{x}$.

A recurrent fuzzy system will be automaton-like if for every vector $(\mathbf{x}, \mathbf{u}) \in X \times U$ there exists a correlated core-position vector, $(\mathbf{s}_j^x, \mathbf{s}_q^u) \in X \times U$, such that the core-position vector, $\mathbf{s}_{w(j,q)}^x = \mathbf{f}(\mathbf{s}_j^x, \mathbf{s}_q^u)$, is also correlated to the mapped vector, $\mathbf{f}(\mathbf{x}, \mathbf{u})$. In this case, the situations prior to the mappings, i.e., $(\mathbf{x}, \mathbf{u})$ and $(\mathbf{s}_j^x, \mathbf{s}_q^u)$, and the results of the mappings, i.e., $\mathbf{f}(\mathbf{x}, \mathbf{u})$ and $\mathbf{f}(\mathbf{s}_j^x, \mathbf{s}_q^u)$, will be similar to one another. The mapping of the core-position vector, $(\mathbf{s}_j^x, \mathbf{s}_q^u)$, that arises from the analysis of a single rule thus approximately describes the mapping of the vector $(\mathbf{x}, \mathbf{u})$. The state graph on which the individual rules are represented thus reflects the dynamic behavior of the recurrent fuzzy system. The recurrent fuzzy system thus behaves similarly to an automaton, e.g. is "automaton-like". However, in contrast to the case for a finite automaton, its states may take on infinitely many values intermediate to the core-position vectors.

The question that needs to be asked now is whether the approach to designing the fuzzy sequential pattern-recognition system considered in Section 4 will lead to an automaton-like recurrent fuzzy

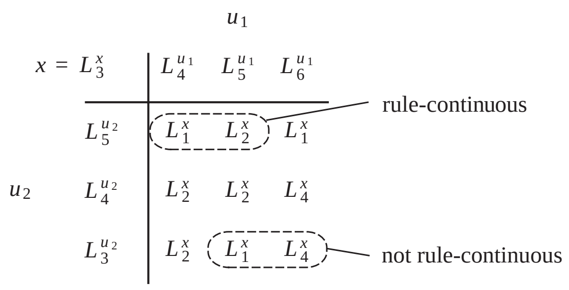
u_1
$x = L_3^x$
$L_4^{u_1}$ $L_5^{u_1}$ $L_6^{u_1}$
rule-continuous
$L_5^{u_2}$ L_1^x L_2^x L_1^x
u_2 $L_4^{u_2}$ L_2^x L_2^x L_4^x
$L_3^{u_2}$ L_2^x L_1^x L_4^x
not rule-continuous

Fig. 11. Part of the rule base describing the behavior of the fuzzy sequential pattern-recognition system as a function of a linguistic state variable, $x = L_3^x$.

system. Since the recurrent fuzzy system has a single state variable, the following theorem[4] from [4] will be applicable:

Theorem 1. *A continuous recurrent fuzzy system that has a single state variable will be automaton-like if, and only if, its rule base is rule-continuous.*

A rule base is said to be "rule-continuous" if changing a linguistic value in the premises to the next-larger or next-smaller value changes no more than a single linguistic value in the conclusions, and changes it to the next-larger or next-smaller value only. This may be readily verified based on the rule base, as the following example demonstrates:

Fig. 11 depicts an extract from a rule base in the form of a rule matrix for a fixed value, $x = L_3^x$, of a linguistic state variable, x, where the rule base contains both rule-continuous and non-rule-continuous subsections. A linguistic input vector of $(u_1, u_2) = (L_5^{u_1}, L_5^{u_2})$ yields a new value, L_2^x, for the linguistic state variable, x. If the value of the linguistic input u_1 is changed to the next-smaller value, i.e., to $u_1 = L_4^{u_1}$, then the new linguistic value of the state variable will be L_1^x instead of L_2^x. Since L_1^x is the next-smaller linguistic value for an initial new value of the linguistic state variable of L_2^x, the rule base is rule-continuous over this partial range. However, this differs from the case for the input vector $(u_1, u_2) = (L_6^{u_1}, L_3^{u_2})$, which leads to a new state value of L_4^x for the linguistic state variable, x: If the value of the linguistic input is changed from $u_1 = L_6^x$ to the next-smaller value, i.e., to $u_1 = L_5^{u_1}$, then the linguistic value of the state variable for the next time step will be L_1^x. Since L_1^x is neither the next-smaller, nor next-larger, nor identical, value of the linguistic state variable for an initial new value of the linguistic state variable of L_4^x, the rule base is not rule-continuous over this particular range.

In fact, the extract from the rule base appearing in Fig. 11 results from the rule base for the sequential pattern-recognition system used for recognizing the course of the curve shown in Fig. 2, based on the procedure described above. Since the rule base is not rule-continuous, the fuzzy sequential pattern-recognition system will not be automaton-like.

[4] A simplified version of the similarity theorem will be stated here. Although the version appearing in [4] may also be applied to special subsets of the entire input/state space, $X \times U$, the criterion stated here is applicable to the entire input/state space only.

This situation arises for all nontrivial fuzzy sequential pattern-recognition systems. For values of the linguistic state variable of L_j^x, where $j \geqslant 3$, a transition, hold, or backstep rule will normally lie immediately adjacent to a reset or end rule in the rule matrix. This implies that the subsequent state, L_1^x, of that reset or end rule will lie immediately adjacent to the subsequent state, L_{j+1}^x, L_j^x, or L_{j-1}^x, of that transition, hold, or backstep rule in the rule matrix. For $j \geqslant 4$, those subsequent states, L_1^x and L_{j+1}^x, L_j^x or L_{j-1}^x, respectively, will differ by more than a single unit, which means that the recurrent fuzzy system is not rule-continuous, and thus not automaton-like, as well.

This initial approach to designing a fuzzy sequential pattern-recognition system will thus normally yield a system that is not rule-continuous, and thus also not automaton-like. However, the "automaton-like" property would be desirable for a pattern-recognition system since its dynamic behavior could then be very easily mathematically predicted. A modification of the fuzzy sequential pattern-recognition system described above allows obtaining "automaton-like" behavior, as will be shown in the next section.

7. Sequential pattern recognition employing hybrid recurrent fuzzy systems

As shown in the immediately preceding section, an automaton-like fuzzy sequential pattern-recognition system cannot be designed using the initial approach described above. As the brief analysis presented above shows, this is due to the fact that the end and reset rules, which reset the pattern-recognition system to its initialization state, and the transition, hold, and backstep rules, which are used during normal operation, lie adjacent to one another in the rule matrix. The rule base is thus not rule-continuous, and the system will thus not be automaton-like.

Designing a fuzzy sequential pattern-recognition system that will behave like an automaton necessitates looking into the fuzzy logic used for analyzing the rules involved. In the case of transition, hold, and backstep rules, it is reasonable to employ fuzzy logic, since the features involved gradually merge into one another in input space. If a degree of recognition, x, does not fall precisely on a singleton position, s_j^x, that is correlated to a pattern primitive, then the result will say that indications of the existence of an adjacent pattern primitive have been detected, but those indications will not be totally convincing. However, in the case of analyzing end or reset rules, the situation will be different: These decisions must be "crisply" taken, since there is no sense in resetting the pattern-recognition "just a bit."

In the following, switching variables will be introduced in order to allow incorporating the desired crisp decisions into a recurrent fuzzy system. Switching variables are described by a pair of linguistic values, such as "on"/"off" or "error"/"OK." The membership functions employed in fuzzification are discontinuous and take on the values 0 or 1 only. Fig. 12 depicts an example of this situation. Choosing membership functions with jump discontinuities provides that one, and only one, linguistic value can have a positive membership value. As in the case of the continuous membership functions, singletons are employed in conjunction with the defuzzification. The recurrent fuzzy system acquires a hybrid nature due to the discontinuous membership functions employed, since it processes both the discrete switching variables and other, continuous, state variables.

The state vector of the fuzzy sequential pattern-recognition system will here be extended by adding a pair of switching variables, one of which is x_2, which takes on the linguistic values $L_1^{x_2} = $ "on" and $L_2^{x_2} = $ "reset," which states whether pattern recognition is running or should be reset. The other,

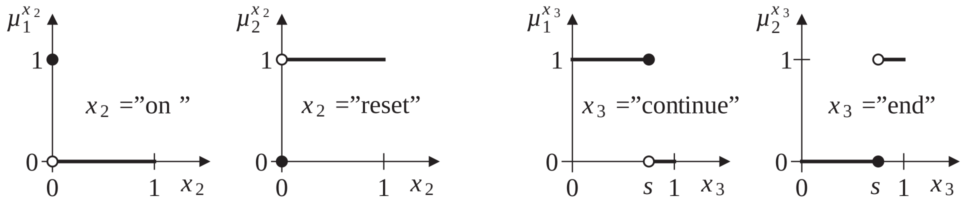

Fig. 12. Membership functions of the switching variables x_2 and x_3, respectively, used in the fuzzification step.

Table 2
Rule base for a fuzzy sequential pattern-recognition system operating in a manner similar to that of an automaton

Rule type	$x_1(k)$	$x_2(k)$	$x_3(k)$	$\mathbf{u}(k)$	$x_1(k+1)$	$x_2(k+1)$	$x_3(k+1)$
Transition rule	$L_j^{x_1}, j < j_{\max}$	On	Continue	$B(L_{j+1}^{x_1})$	$L_{j+1}^{x_1}$	On	Continue
Hold rule	$L_j^{x_1}, j < j_{\max}$	On	Continue	$B(L_j^{x_1})$	$L_j^{x_1}$	On	Continue
Backstep rule	$L_j^{x_1}, j > 1$	On	Continue	$B(L_{j-1}^{x_1})$	$L_{j-1}^{x_1}$	On	Continue
Reset rule I	$L_j^{x_1}$	On	Continue	$A(L_j^{x_1})$	$L_1^{x_1}$	Reset	Continue
Reset rule II	Arbitrary	Reset	Continue	U	$L_1^{x_1}$	On	Continue
End rule I	$L_{j_{\max}}^{x_1}$	On	Continue	$B(L_{j_{\max}}^{x_1})$	$L_{j_{\max}}^{x_1}$	On	End
End rule II	Arbitrary	Arbitrary	End	U	L_1^{x}	On	Continue

x_3, which takes on the linguistic values $L_1^{x_3} =$ "continue" and $L_2^{x_3} =$ "end," states whether pattern recognition will continue or end, in which case all pattern primitives of the pattern sought have appeared in the correct order and have been recognized. Their associated membership functions employed in conjunction with the fuzzification are depicted in Fig. 12. The threshold value, at which the jump in the membership functions occurs, has, in the case of the reset variable, x_2, been chosen close to zero. Reliable resets will thus occur as soon as the pattern detected differs from the pattern to be recognized. In the case of the membership function of the end variable, x_3, the location, s, of the jump discontinuity will determine the truth value of the uppermost linguistic recognition stage, $x_1 = L_{j_{\max}}^{x_1}$, at which recognition of the pattern will be regarded as having been concluded. Once again, singletons are employed in conjunction with the defuzzification, in this case, singletons at the positions $s_1^{x_{2/3}} = 0$ and $s_2^{x_{2/3}} = 1$.

The rules for this new pattern-recognition system are obtained by modifying the rules of the initial approach, which are listed in Table 1. During normal operation, i.e., for $x_2 =$ "on" and $x_3 =$ "continue," the transition, hold, and backstep rules remain unchanged. The old reset rule is split into two new rules: If a reset condition should occur during operation, then Reset Rule I will set the variable x_2 to the value $x_2 =$ "reset," while the state variable, x_1, will be set to an arbitrary value, e.g., $L_1^{x_1}$. In the second step of the reset, Reset Rule II will cause the system to be reset to its initialization state, i.e., $x_1 = L_1^{x_1}$, $x_2 =$ "on", $x_3 =$ "continue". The former end rule will also be split into two new rules: If the counter variable, x_1, reaches the value $x_1 = L_{j_{\max}}^{x_1}$ during operation, End Rule I is employed. End Rule I resembles a sort of hold rule for the state $x_1 = L_{j_{\max}}^{x_1}$, for which, however, the end variable will be set to the value $x_3 =$ "end". In the second step, End Rule II will cause the system to be set to the initialization state, i.e., $x_1 = L_1^{x_1}$, $x_2 =$ "on," $x_3 =$ "continue". The new rules for the hybrid fuzzy sequential pattern-recognition system are summarized in Table 2.

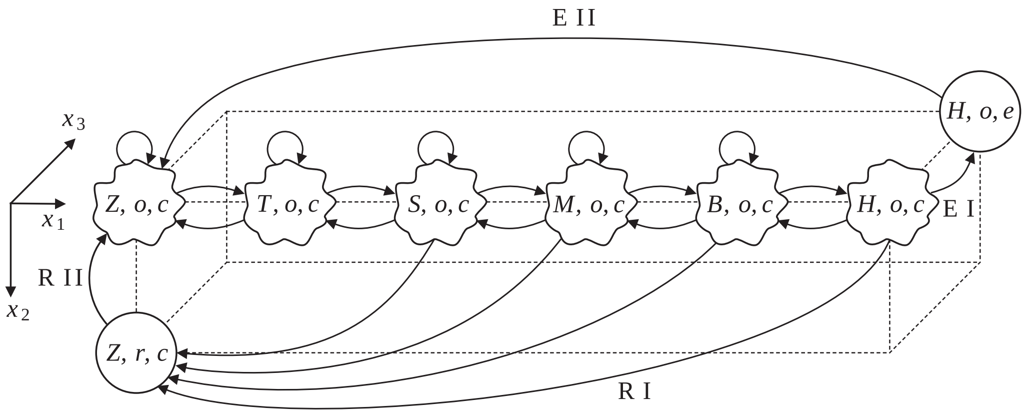

Fig. 13. Transition graph for a sequential pattern-recognition system based on a recurrent fuzzy system that uses switching variables. The new types of rules, i.e., End Rule I (E I), End Rule II (E II), Reset Rule I (R I), and Reset Rule II (R II), are the only rules that are marked on the transition pointers.

Once again, a state graph for the recurrent fuzzy system may be prepared, based on the rule base, and is shown in Fig. 13. The state graph is depicted against a background containing the outline of a rectangular solid indicating the three-dimensional state space, X, of the recurrent fuzzy system. The recognition states that are taken on during operation, i.e., for $x_2 =$ "on," combined with $x_3 =$ "continue," are indicated by wavy nodes due to their fuzzy interpretations. The other states are indicated by circular nodes, in keeping with their sharp natures. Several states, such as (T, r, c) or (S, r, e), on which no transition arrows terminate, and all of which lead to the initialization state, have not been shown in the interest of greater clarity. We then have that:

Theorem 2. *A hybrid fuzzy sequential pattern-recognition system will either be an automaton or be automaton-like if the rule base is rule-continuous with respect to the components x_1 in the vicinities of the transition, hold, and backstep rules and of End Rule I.*

Proof. In the proof, all of the various types of transition rules will be considered individually in turn.

End Rule II is the only type of rule that processes those linguistic inputs for which $x_3 =$ "end," according to Table 2. The truth value for End Rule II will be either 0 or 1, since the conditions for x_1, x_2, and $\mathbf{u}$ in the premises will always be fully met and the condition for x_3 will be either fully met or not met at all. Thus, if this rule becomes active, it will be the sole active rule, in which case, the system will behave precisely like an automaton, since it will also process just a single rule. The decision regarding the end of the pattern recognition will thus be a sharp decision.

Reset Rule II may be considered in a manner similar to that for End Rule II, since, except for End Rule II, it is the only type of rule that processes linguistic inputs for which $x_2 =$ "reset." As in the case of End Rule II, the truth value for Reset Rule II will be either 0 or 1. If it becomes active, it will be the sole active rule, in which case, the system's state will be reset to the initialization state, $(L_1^{x_1}, o, c)$, and, in this case as well, the system will be an automaton.

Unlike End Rule II and Reset Rule II, Reset Rule I may be active simultaneously with other rules. Its truth value may thus also take on values intermediate between 0 and 1. Weighting the results of the individual active rules then yields a new value of x_2 at time $k+1$ exceeding the core position $s_1^{x_2}=0$, since Reset Rule I will be active to a certain degree. Due to the jump discontinuities in the membership functions of x_2 shown in Fig. 12, Reset Rule II, or, in special case, End Rule II, will be the sole active rule at the next time step, as the foregoing considerations show. Thus, $\mathbf{x}(k+1)$ will then yield the subsequent state $\mathbf{x}(k+2)=(L_1^{x_1},o,c)$, regardless of the value of $x_1(k+1)$ and the reset procedure will have been successfully concluded. The operation of the pattern recognition will thus be unaffected if a particular value of $x_1(k+1)$ should violate its automaton-like property, as defined in [4] during this single step. In this case as well, the hybrid fuzzy sequential pattern-recognition system will be an automaton, for which the state variable x_1 may also be intermittently assigned an arbitrary value.

Transition, hold, or backstep rules become active during operation only, i.e., for $x_2=$ "on," combined with $x_3=$ "continue," only. They also change just the first component of the state vector, which is why x_1 will, in this case, be the sole dynamic variable. Investigation of the state vector's similarity to an automaton may thus be confined to considering its first component, x_1. Since the rule base is rule-continuous with respect to the component, x_1, in the vicinities of the transition, hold, and backstep rules, according to the premise of Theorem 2, it follows from the similarity theorem [4] that the system will also be automaton-like in those vicinities.

End Rule I is the last type of rule to be treated. If this is the only rule active, then the system will both be an automaton and be automaton-like [4]. In all other cases, it may, for the purpose of investigating the system's similarity to an automaton, be treated as an imaginary hold rule for the state $L_{j_{\max}}^{x_1}$, as will be shown below:

Assume that the system employs a hold rule for the state $L_{j_{\max}}^{x_1}$ instead of End Rule I. Such a system will be automaton-like if a combination of transition, hold, or backstep rules is active, as has been shown above. For each of the state vectors, $\tilde{\mathbf{x}}(k)$ and $\tilde{\mathbf{x}}(k+1)=\mathbf{f}(\tilde{\mathbf{x}}(k),\mathbf{u}(k))$, before and following the mapping, respectively, there will thus be correlated core-position vectors, $\mathbf{s}_j^x$ and $\mathbf{s}_w^x=\mathbf{f}(\mathbf{s}_j^x,\mathbf{s}_q^u)$, respectively, with a core-position vector, $\mathbf{s}_q^u$, correlated to $\mathbf{u}(k)$. Since, in this case, exclusively transition, hold, and backstep rules are active, the third component, $\tilde{x}_3(k+1)$, of $\tilde{\mathbf{x}}(k+1)$ will coincide with the core position $s_1^{x_3}$ of the linguistic value "continue," i.e., $\tilde{x}_3(k+1)=s_1^{x_3}$. The core-position vector, $\mathbf{s}_w^x$, correlated to $\tilde{\mathbf{x}}(k+1)$ thus also has a third component, $s_{w_3}^{x_3}=s_1^{x_3}$.

The same core-position vectors, $\mathbf{s}_j^x$ and $\mathbf{s}_w^x$, may also be employed for proving the system's similarity to an automaton in the case where End Rule I is employed. In order to prove that similarity, consider the same state vector $\mathbf{x}(k)=\tilde{\mathbf{x}}(k)$ as above and its mapped vector, $\mathbf{x}(k+1)=\mathbf{f}(\mathbf{x}(k),\mathbf{u}(k))$. Like $\tilde{\mathbf{x}}(k)$, the vector $\mathbf{x}(k)$ is correlated to the core-position vector $\mathbf{s}_j^x$. The new state vector, $\mathbf{x}(k+1)$, differs from $\tilde{\mathbf{x}}(k+1)$ only in its third component, since End Rule I and the hold rule differ only in their third component of their conclusion portions. We thus have that $\tilde{x}_1(k+1)=x_1(k+1)$ and $\tilde{x}_2(k+1)=x_2(k+1)$. Furthermore, $x_3(k+1)$ lies between the core positions $s_1^{x_3}$ and $s_2^{x_3}$, and can always be rounded off to $s_1^{x_3}$ during the search for correlated core-position vectors. $\mathbf{s}_w^x$ will thus also be correlated to $\mathbf{x}(k+1)$. Like the hold rule, End Rule I thus also leads to automaton-like behavior. □

Rule bases confined to transition, hold, and backstep rules will normally be rule-continuous. However, there are two situations in which rule-continuity may be violated. Under the first situation, transition rules and backstep rules might be immediately adjacent to one another in the rule matrix for a fixed linguistic value, L_j^x. Under the second situation, it might happen that both a backstep rule for the value L_j^x of the linguistic state variable and a transition rule for the value L_{j+1}^x of the linguistic state variable apply for a fixed linguistic input vector, $\mathbf{L}_q^u$, i.e., there exists an $\mathbf{L}_q^u \in B(L_{j-1}^x) \cap B(L_{j+2}^x) \neq \emptyset$. In either case, it will usually be possible to separate the rules involved within the rule base by introducing another value of the linguistic state variable lying between L_j^x and L_{j+1}^x. The recurrent fuzzy system will then be rule-continuous and automaton-like in these regions as well.

None of the aforementioned exceptional cases will occur if the membership functions shown in Fig. 10 are employed in the case of the example of the breakout-recognition system. Theorem 2 is immediately applicable to a hybrid fuzzy sequential pattern-recognition system using these membership functions. In areas, where solely transition, hold, and backstep rules are active and the only dynamic state variable is x_1 the fuzzy pattern-recognition system resembles a one dimensional, terminated, rule-continuous, standardized, recurrent fuzzy system as defined in [15]. Thus Theorem 5 appearing in [15] also provides that its state variable will converge to a fixed value, provided that solely transition, hold, and backstep rules are active and the value of the input variable remains constant. Furthermore, chaotic dynamics that can occur in other recurrent fuzzy systems [4] are thus precluded in this case according to Theorem 10 of [15].

If the sampling rate is too low, i.e., if the sampling time is approximately equal to the duration of a pattern primitive, the "curcurve-problem" described in Section 2 may arise. At a suitably high sampling rate, the same pattern primitives will occur over several consecutive time steps, e.g., the input signal will be similar to the "ccccuuuurrrcccccuuuurrrvvvvveeee" appearing in the example of the sampled signal, "curcurve". Using this sampling rate, the first occurrence of a "c" in the second group of characters, "c," will initiate the reset. The next character, another "c", will be ignored, since the reset procedure takes two time steps, instead of one, under this new setup. Following the reset, the next character, the third "c" in the second group of pattern primitives, "c", will be evaluated and treated as the commencement of a new search pattern, "curve". Thus, the pattern "curve" will be found in the suitably rapidly sampled pattern, "curcurve".

We shall now summarize the benefits provided by fuzzy sequential pattern-recognition systems employing hybrid recurrent fuzzy systems: Firstly, recurrent fuzzy systems are based on rules that may also be employed by conventional sequential pattern-recognition systems. The rules involved are thus immediately interpretable. Secondly, extraction of the features of a time series systematically follows from its linguistic description and fuzzification. Thirdly, the fuzzy state yields both a statement regarding the number of recognized pattern primitives, as in the case of a conventional automaton, and whether, and to which extent, the next feature begins to appear in the input variables, since its state variable may also take on intermediate values. Fourthly, they provide that, in many cases, their system behaviors will be automaton-like, or that they are finite automata.

8. Summary and conclusions

In addition to being suitable for use in pattern-recognition tasks that utilize static functions, fuzzy systems may also be used in sequential pattern recognition. Recurrent fuzzy systems that feed values of their output values back to their input with a time delay are ideal choices for such use. Their configuration follows immediately from the linguistic description of the particular series of pattern primitives to be recognized and the verbal description of the behavior of a sequential pattern-recognition system. This sort of system has proven its worth in industrial use as a monitoring system on continuous-casting systems used in the steel industry. The modified version of a recurrent fuzzy system for use in sequential pattern recognition that has also been presented here provides the same high degrees of transparency and functionality provided by the initial version. Moreover, the former's dynamic behavior may, in many cases, be approximated by that of a finite automaton and thus readily understood.

References

[1] J. Adamy, Device for early detection of runout in continuous casting, EP Priority Date 03.04.1995, European Patent EP 0 819 033 B1, 1998, US Patent 5,904,202, 1999.[5]

[2] J. Adamy, Breakout prediction for continuous casting by fuzzy mealy automata, Proc. of EUFIT'95, Aachen, 1995, pp. 754–759.

[3] J. Adamy, J. Freitag, S. Lorenz, Process for parametering a fuzzy automaton that compares a measurement system to a pattern signal, European Patent EP 0 941 504 B1, 2001, US Patent US 6,345,206 B1, 2002.

[4] J. Adamy, R. Kempf, Regularity and chaos in recurrent fuzzy systems, Fuzzy Sets and Systems 140 (2) (2003) 259–284.

[5] J. Adamy, A. Yousif, Controller with dynamic discrete-time fuzzy-systems, German Patent application DE19734711, 1997, European Patent EP 1 002 260 B1, 2002.

[6] P. Bardet, A. Leclercq, M. Mangin, B. Sarter, Contrôle du processus de coulée continue de brames à la Sollac, Rev. Metall. - CIT 80 (1983) 303–312.

[7] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.

[8] J.C. Bezdek, K.P. Sankar, Fuzzy Models for Pattern Recognition: Methods That Search for Structures in Data, IEEE, New York, 1992.

[9] R.O. Duda, P.E. Hart, G.E. Stork, Pattern Classification, Wiley, New York, 2001.

[10] R. Gonzalez, M. Thomason, Syntactic Pattern Recognition, Addison-Wesley, Reading, MA, 1978.

[11] V. Gorrini, H. Bersini, Recurrent fuzzy systems, Proc. 3rd IEEE Conf. on Fuzzy Systems, Orlando, FL, 1994, pp. 193–198.

[12] M. Hormel, M. Schwartz, Trainable process monitoring system applied to sticker detection in continuous casting, Proc. of EUFIT'98, Aachen, 1998, pp. 1458–1462.

[13] S.Y. Hwang, H.-S. Lee, J.J. Lee, General fuzzy acceptors for syntactic pattern recognition, Fuzzy Sets and Systems 80 (1996) 397–401.

[14] S. Itoyama, et al., Process of and apparatus for continuous casting with detection of possibility of break out, US Patent 4,949,777, 1990.

[15] R. Kempf, J. Adamy, Equilibria of recurrent fuzzy systems, Fuzzy Sets and Systems 140 (2) (2003) 231–257.

[16] V. Krebs, E. Schäfers, Dynamische Fuzzy Systeme Zur Qualitativen Prozeßmodellierung, GMA-Conference on 'Computational Intelligence', Berlin, VDI-Report 1381, 1998, pp. 115–135.

[17] A. Koski, M. Juhola, M. Meriste, Syntactic recognition of ECG signals by attributed finite automata, Pattern Recognition 28 (12) (1995) 1927–1940.

[18] S. Kubota, A. Koyama, Y. Sasabe, H. Miki, A real-time expert system applied to the mold bath level control of a continuous caster, Sumitomo Search 50 (1992) 51–58.

[19] M. Langer, M. Arzberger, New approaches to breakout prediction, Steel Times Internat. 26 (10) (2002) 23–26.

[20] J.M. Lu, K.J. Lin, C.H. Kou, W.C. Chien, Sticker breakout theory and its prediction in slab continuous casting, Proc. 76th Steelmaking Conference, Dallas, TX, 1993, pp. 343–353.

[21] K.C. Mills, T.J.H. Billany, A.S. Normaton, B. Walker, P. Grieveson, Causes of sticker breakout during continuous casting, Ironmaking Steelmaking 18 (4) (1991) 253–265.

[22] H. Niemann, Methoden der Mustererkennung, Akademische Verlagsgesellschaft, Frankfurt a. M., 1974.

[5] Erratum: Reference [2] in our article [4] and reference [1] in our article [15] should read as reference [1] in this article.

[23] M. Niemann, J. Adamy, H.-J. Nitsche, Modular mould level control for continuous casting plants, Metallurgical Plant and Technology Internat. 20 (5) (1997) 92–98.
[24] M. Niemann, J. Adamy, H.-J. Nitsche, Method and device for casting a strand from liquid metal, German Patent DE 196 33 738 C, 2002, Italian Patent IT 0001293830 B, 2002, US Patent US 5,915,456, 1999.
[25] R.P. Nikhil, Pattern Recognition in Soft Computing Paradigm, World Scientific, Singapore, 2001.
[26] A. Nürnberger, R. Kruse, A neuro-fuzzy approach to optimize hierarchical recurrent fuzzy systems, Fuzzy Optim. Decision Making 1 (2002) 221–248.
[27] M.A. Patten, A. Klein, M.M. Wolf, Advances breakout prevention system custom-made to product mix, Proc. 74th Steelmaking Conference, Washington, DC, Vol. 74, 1991, pp. 553–560.
[28] W. Pedrycz, Fuzzy sets in pattern recognition: accomplishments and challenges, Fuzzy Sets and Systems 90 (1997) 171–176.
[29] K. Peeva, Fuzzy acceptors for syntactic pattern recognition, Internat. J. Approx. Reason. 5 (1991) 291–306.
[30] B.-M. Pfeiffer, J. Jäkel, A. Kroll, C. Kuhn, H.-B. Kuntze, U. Lehmann, T. Slavinski, V. Tews, Erfolgreiche Anwendungen von Fuzzy Logik und Fuzzy Control (Teil 1), Automatisierungstechnik 50 (10) (2002) 461–471.
[31] E. Schäfers, V. Krebs, Dynamic Fuzzy Systems for Qualitative Process Modeling: Principles of a New System Theory, at 8/99, 1999, pp. 382–389.
[32] E. Sowka, P. Schulze-Diekhoff, J. Harder, F. Munscher, G. Beirer, Breakout avoidance system, BASYS, for continuous slab casting, Iron Steel Eng. 76 (5) (1999) 30–34.
[33] H. Surmann, M. Maniadakis, Learning feed-forward and recurrent fuzzy systems: a genetic approach, J. Systems Archit. 47 (2001) 649–662.
[34] T. Tanaka, H. Endo, N. Kamada, S. Naito, H. Kominami, Trouble forecasting system by multi-neural network on continuous casting process of steel productions, in: T. Kohonen, et al., (Eds.), Artificial Neural Networks, Elsevier, Amsterdam, 1991, pp. 835–840.
[35] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading MA, 1974.
[36] A. Tsuneoka, W. Ohasi, S. Ishitobi, T. Kataoka, M. Tenma, Measurement and control system of solidification in continuous casting mold, Proc. ISS-AIME Steelmaking Conference, Detroit, MI, 1985, pp. 3–10.
[37] Fuzzy logic and fuzzy control—Terms and definitions, VDI/VDE-Richtlinien, VDI/VDE 3550 Blatt 2, Beuth Verlag GmBH, Berlin, 2000.
[38] J. Zhang, Recurrent neuro-fuzzy networks for the modeling and optimal control of batch processes, Proc. Jt 9th IFSA World Congress and the 20th NAFIPS International Conference, Vancouver, B.C. 2001, pp. 523–528.