Acta Universitatis Sapientiae
Electrical and Mechanical Engineering, 3 (2011) 5-14

Action Descriptive Strings

Péter Megyesi[1], Sándor Molnár[2]

Department of Telecommunications and Media Informatics,
Faculty of Electrical Engineering and Informatics,
Budapest University of Technology and Economics, Budapest, Hungary
e-mail: megyo@fazekas.hu[1], molnar@tmit.bme.hu[2]

Manuscript received December 20, 2011; revised January 16, 2012.

Abstract: An outcome of a measurement frequently contains too many and redundant elements so extracting similar patterns can be a hard task. This paper presents an idea to convert time streaming measurement results into a special string format and to use existing motif finding methods for further analysis. We have created the Action Descriptive String (ADS) which is a projection of series of events. Although these strings are not an accurate description of the actual events they had been defined from, they are helpful in finding typical occurrences of short term events and comparing two series of events.

Following the presentation of the existing motif finding methods and the benefit of applying them for the Action Descriptive Strings, we give the idea of converting measurement results into ADS format. The method for finding frequently occurring patterns in the converted string format is also presented. We discuss the possibility of emulating long term events by a series of the extracted typical short term patterns. We have implemented an algorithm which is able to score the similarities between two Action Descriptive Strings. This algorithm is used for finding the most similar typical pattern for an arbitrarily given ADS. Since the scoring scheme of the algorithm is highly dependent on the measured phenomenon, we also present a process for adjusting the scoring values to a given measurement.

Keywords: String matching, motif finding, measurement analysis.

1. Introduction

Converting time serial measurement results into appropriate short string format can help us applying existing string based algorithms for further analyze.

This motivation led us to create a special string format (Action Descriptive Strings) from these raw measurement results.

Motif finding is widely used method in bioinformatics and has a deep literature that we can rely on. In bioinformatics the algorithms tend to find similarities in protein chains, typically in amino acid. These algorithms usually use a simple string input which contains the markings of the four proteins DNA is built from: A for Adenine, T for Thymine, G for Guanine and C for Cytosine. MEME suite is a comprehensive tool for discovering motifs in a group of related DNA or protein sequences [1].

Similar architectures are presented in [2], [3] and [4] for using string matching algorithms for signature generation in various network appliances. In these cases one byte represents a unit in the sequence therefore there are 256 different symbols.

The Action Descriptive Strings (ADS) can be used for representing any kind of time series measurement result. Using these strings and the algorithms presented in this paper the measurements can be further analyzed from a different point of view. We have implemented the ADS in a general way that can be suitable for various measurement types.

In the next section the general structure of the Action Descriptive Strings is presented. The idea of extracting typical patterns from converted measurements results is discussed in Section 3. Section 4 presents an algorithm which is able to suit a series of typical patterns for any given measurement result stream. The algorithm we implemented for scoring the similarities between two ADS is given in Section 5. Section 6 presents the idea of adjusting the scoring algorithm for unique measurement types. Finally, in Section 7 a summary of our work is found where we present a few possible utilizations for the Action Descriptive Strings.

2. Action Descriptive Strings

In an Action Descriptive String there are two types of characters: the action characters and the time delimiter characters. Although the time delimiter character can be chosen arbitrarily, we chose the “Z” characters for separating equal time intervals in an ADS. The other characters refer for one type of action. Thus an example an example for them looks like the following:

AZAZABZABZACZAZAZ

Since this example contains seven “Z” characters, this ADS represents a seven-time-unit-long scenario. This means that in this scenario an “A” type of action occurred in every seven time unit, a “B” type of action happened in the third and the fourth, and a “C” type occurred in the fifth time unit.

During the conversation of measurement results the data must be separated to individual sources. Thus in case of an aggregated measurement the first step must be a discrimination of the sources (typically location based). In the next step every different action must be investigated for every time unit. If an action occurred in the given time unit we write its character into the ADS than we close the given time unit by the “Z” character.

The definition of the actions is a heuristic process which is highly dependent on the measurement itself. They can be simple event occurrences (for example if a motorcycle has crossed the road or a costumer has bought milk) or a limit for event occurrences (for example if the number of cars that crossed the road is greater than 100). Limit bands are also usable in event definition for detailed resolution but since the idea is to simplify the measurement results these bands should be minimized.

The choice of the used time resolution is also has to be adjusted for the given measurement type. Too short time units can result to an output where typical patterns are not extractable. On the other, hand using a too long time unit could hide the differences between distant inputs.

```
1223378304 ABZABZABZABZABZABZABZABZABZABZ
1223378306 HIZHIZHIZHIZHIZHIZHIZHIZHIZHIZ
1223378809 FZFZFZFZFZFZFZFZFZFZ
```

Figure 1: The format of the input ADS file.

Taking these into account the format of the required input file is given in *Figure 1*. As the figure shows the input should also contain the UNIX timestamp of the beginning of the different sources’ action which will be used during the long term event emulation.

3. Extracting typical patterns

During the extract of typical patterns we tend to find substrings which are frequently occur in the input. In this procedure we search for fixed time length patterns between lower and a higher limit. The reason for the lower limit is that too short patterns are not describing stable event series. On the other hand, the longer a pattern the lesser it would occur in the input stream. Thus these limits have to be defined considering the time input’s time resolution.

As a first solution we have inspected bioinformatics problems. We have examined the possibility of applying the same architecture presented in [2] with a different preprocessing method. In this paper the authors present a framework using Glam2 for signature generation. Glam2 is a software package for finding motifs in sequences, typically amino-acid or nucleotide sequences [5]. Glam2Scan is a part of the Glam2 software package which can find matches in a sequence database to a motif discovered by Glam2 [6]. Glam2Scan gives a score for each match indicating how well it fits the given motif so it also could be the base an approximately match algorithm during a long term event emulation.

However, after a long investigation process we have found that we can't use this technique for the general architecture of Action Descriptive Strings. The main reason behind this is that bioinformatics algorithm use alphabet where the roles of the characters are the same. In ADSs the time delimiter ("Z") character has completely different meaning than the others. Moreover, in our case we would like to have a method to define suitable replacements for certain event types therefore defining subclasses where the events are similar with each other and distant from the others.

Thus we implemented a unique algorithm which is able to extract typical patterns from the input. The key point in the algorithm is to split the Action Description Strings along the time delimiter ("Z") characters. That way we make sure that the individual time units' activities won't be corrupted. For preprocessing the algorithm does two things. First, it filters out the long idle periods (multiple consecutive "Z" characters) in the sources' activities thus the algorithm won't give back patterns in which the idle period is longer than the actual activity. Secondly, for symmetric ADS patterns (where the same characters occur in every time unit) the algorithm recalculates the real number occurrences. For example, in case of a 200 time-unit-long "*AZ*" run the occurrence for the five-time-unit-long "*AZAZAZAZAZ*" should be 40 not 196.

In the algorithm two limits have to be declared: a *hard limit* and *soft limit*. First, the tool counts the occurrences for every occurring ADS substring which has the given time length. Then the algorithm calculates the most similar ADS for every pattern below the *soft limit* using the scoring mechanism presented in Section 5 and increase the result's occurrence by one. After that, the ADS substrings occurring more than the *hard limit* are added into the typical patterns pool.

The choice of the limits is dependent on the result we would like to achieve. For example, setting the two limits to the same value will avoid the usage of the scoring mechanism and result in only the ADSs which occurrence is greater than then the given limit. On the other hand, setting the *soft limit* to 1 will give the most detailed result but it can significantly increase the run time of the algorithm.

4. Long term event emulation

The main idea behind the emulation of long term events is to substitute the entire stream of one source by a series of typical patterns. In order to do that, we have the input Action Descriptive String file and the database of the extracted typical patterns. The input ADS can be one line from the measurement result or can be generated artificially. The second method allows us to emulate arbitrary event type that would otherwise not occur in real measurements. In practice it means that we need an algorithm which is able to cover a source's entire Action Descriptive String with the extracted typical patterns' ADSs. Moreover, the algorithm must calculate the accurate timing information.

The first part of the implemented algorithm is a search for full-matching typical patterns in the input ADS. During this process two rules should be kept. First, we have to start the search with the longer patterns. During the emulation process we prefer the usage of longer term typical patterns since we consider their activity more stabile. With this action we make sure that we use the longer scenarios as much as possible. The second rule is that if we find a full-matching user pattern somewhere in the input ADS we have to switch that substring to only time delimiter ("Z") characters. That way we guarantee that the time units in the input stream will be covered by only one pattern. However, we have to leave the time delimiter characters in the ADS in order to properly calculate the timing information.

Since an arbitrary Action Description String can unlikely be covered by only typical patterns we have implemented an approximate matching part for the algorithm as well. During this procedure the algorithm uses a scoring scheme introduced in the next section which is able to find the most similar typical pattern for any given ADS.

After full-matching the remaining ADS may contain many consecutive time delimiter ("Z") characters. Since this "Z" runs means an inactive period or that it has been previously covered by a full-matching typical pattern, as a next step the algorithm splits the remaining string along three or more consecutive "Z" characters. The last step of the algorithm searches for the most similar typical pattern for every remaining substring after the split. These time period in the input stream will be emulated by the same action as the most similar typical pattern describes. If a reaming substring is longer than the higher time limit of the typical patterns the algorithm calculates the score for every prefix from the lower time limit to the higher. The one with the best result score will be emulated by the most similar pattern than the algorithm removes that ADS from the substring and repeats the approximate matching procedure.

5. The ADS scoring algorithm

When we were designing the ADS scoring algorithm the following rules were laid down:

1) If we compare an A sting to a B sting, the returned score must be less than or equal to score the algorithm returns comparing the A string with itself.
2) The equality must only stand if the same characters with the same amount are in both A and B.
3) The algorithm must inspect the time length of the ADSs and give lesser score if it differs.
4) We must have a way of setting unique values for which action types are suitable substitutions for each other and which are completely excluded.

Taking these considerations into account, we have defined a scoring matrix labeling its rows and columns with the defined types' characters. We also add the "X" character which will refer to no action. If we substitute an "A" character from the first string to a "B" character in the second string the score under the "A" row and "B" column will be added to the total score. Firstly, the algorithm concatenates "X" characters to the shorter string until both of them contain the same amount of characters. After this, a dynamic programming algorithm finds the best substitution solution for the characters calculating the maximal possible score [7], [8]. As the last step the algorithm modifies the given score with a divider if the time lengths of two strings differ.

In order to get the most ideal values of the scoring matrix and the length modifier we created a test database of Action Descriptive Strings. In contrast with the typical patterns this artificial database contains only ADSs in which the actions are the same in every minute. We integrated every variation of minimum four maximum ten minute long scenarios which contains maximum 4 type of action simultaneously. In the following section we will present a method for proper adjustment of the scoring values using the test database.

6. Adjusting scoring values

In this example we use Action Descriptive Stings which are made of four different characters: *"A"*, *"B"*, *"C"* and *"D"*. The initial scoring matrix is shown in *Table 1*. The scoring values were set up as following. "A" is the least significant action and we prefer to substitute it primarily to "B" and secondly to no action ("X"). Substitute "A" to "C" or "D" will decrease the score as their action is considered to be too distant. In case of the action marked by "B" the suitable substitutes are primarily "C" and secondly "A". For "C" we prefer "D" and “B” while for “D” the suitable substitutions are “C” and “B”. If one action is replaced by itself five points are added to the final score.

All the values in the first row of *Table 1* are negative. This step is required for keeping the first rule since positive values would result in more score in case of adding more extra action. The values are different thus in case of an extra action the least significant actions will be preferred.

Table 1: An initial scoring matrix.

	X	A	B	C	D
X	0	-2	-3	-4	-5
A	2	5	3	-1	-2
B	0	2	5	3	1
C	-2	0	2	5	3
D	-5	-1	1	3	5

Table 2: Results for the initial scoring values.

Rank	Score	Relative score	Time unit	ADS
0	30	1	5	AZAZAZAZABZ
1	25	0.83	5	AZAZAZAZAZ
2	19	0.63	5	BZBZBZBZBZ
3	18	0.6	5	ABZABZABZABZABZ
4	12	0.4	5	ACZACZACZACZACZ
5	6	0.2	5	ADZADZADZADZADZ

Table 2 shows the five most similar ADS from the test database for the input *AZAZAZAZABZ*. This is a five-time-unit-long event which contains the action “A” in every time unit and the action “B” in the last one. The first row (rank 0) contains the result comparing the input string with itself. As it can be seen in *Table 2*, this result is 30 since there are six action characters in this ADS thus the final score is six times five.

Although the results show what can be previously expected, the actual rankings and the given points can be further investigated. For example, the second and the third result show that these scoring values prefer the substitution of an entire action to another than using both of them. An adjustment procedure for real measurement result can be inspected in many ways which are highly dependent on the phenomenon. For example, if the actions are fairly different from each other the scoring values for the preferred substitutions should be decreased significantly.

An example for this type of scoring matrix is presented in *Table 3*. The difference between this scoring matrix and the initial one is that the positive values for the substitutions are divided by ten. That way, substitute one character for a different one will result in lesser score. The results for the same input as in the previous test is given in *Table 4*. In this case the ADS where both “A” and “B” occurred in every time unit is the second most similar to input with the same score while the pattern containing only “B” got significantly less score. If this similarity is closer to the realty than the first one the modified scoring matrix should be used.

Table 3: Example for modified scoring matrix.

	X	A	B	C	D
X	0	-2	-3	-4	-5
A	0.2	5	0.3	-1	-2
B	0	0.2	5	0.3	0.1
C	-2	0	0.2	5	0.3
D	-5	-1	0.1	0.3	5

As a last example we let the algorithm to test Action Descriptive Strings which time length differs from the input. In these cases the calculated score from the scoring matrix is divided by a constant. The results using 1.2 as the length divider are presented in *Table 5*. As the results show the four and six time-unit-long variants of the previously best two results appeared in the array.

Table 4: Results for the modified scoring matrix.

Rank	Score	Relative score	Time unit	ADS
0	30	1	5	AZAZAZAZABZ
1	25	0.83	5	AZAZAZAZAZ
2	18	0.6	5	ABZABZABZABZABZ
3	9.3	0.31	5	ACZACZACZACZACZ
4	6.4	0.21	5	BZBZBZBZBZ
5	5.1	0.17	5	ADZADZADZADZADZ

Table 5: Results using different time lengths.

Rank	Score	Relative score	Time unit	ADS
0	30	1	5	AZAZAZAZABZ
1	25	0.83	5	AZAZAZAZAZ
2	21	0.7	6	AZAZAZAZAZAZ
3	18	0.6	5	ABZABZABZABZABZ
4	16.83	0.56	4	AZAZAZAZ
5	16.08	0.53	4	ABZABZABZABZ
6	10.83	0.36	6	ABZABZABZABZABZABZ

7. Conclusion

We have presented Action Descriptive Strings (ADS) which is a projection of real measurement results. By this conversation we are able to use existing motif finding methods for further analyzing a time stream measurement.

We have given methods for finding short term typical patterns in the input stream and use them for emulating long term activities. Both of these processes use an algorithm which can score the similarities between two Action Descriptive Strings. This paper contains an example for adjusting the scoring values of the ADS scoring algorithm using an artificially created pattern database.

An example for the application of the Action Descriptive Strings can be found in [7]. In [7] the author uses the same architecture for describe network traffic and determine the typical way of how users are using the Internet.

Acknowledgements

The authors would like to thank Géza Szabó, coworker of Ericsson Hungary Ltd. for providing support for this project.

References

[1] MEME suite: http://meme.sdsc.edu/meme/

[2] Szabó, G., Turányi, Z., Toka, L., Molnár, S., Santos, A.: "Automatic Protocol Signature Generation Framework for Deep Packet Inspection", *VALUETOOLS 2011, ENS, Cachan, France*, May 16-20, 2011.

[3] Ye, M., Xu, K., Wu, J., and Po, H.: "Autosig-automatically generating signatures for applications", in *CIT (2)- IEEE Computer Society*, 2009, pp. 104–109.

[4] Conrad, E., "Detecting Spam with Genetic Regular Expressions", http://www.sans.org/reading_room/whitepapers/email/detecting_spam_with_genetic_regular_expressions_2006.

[5] Glam2 manual: http://meme.sdsc.edu/meme/doc/glam2_man.html.

[6] Glam2Scan manual: http://meme.sdsc.edu/meme/doc/glam2scan_man.html.

[7] Megyesi, P., "Matching Algorithm for Network Traffic Descriptive Strings", *Scientific Students' Associations Conference, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics*, Nov 2011.

[8] "Dynamic Programming", http://www.cs.berkeley.edu/~vazirani/algorithms/chap6.pdf.