(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 September 2007 (20.09.2007)

PCT

(10) International Publication Number
WO 2007/106365 A2

(51) International Patent Classification:
G06F 7/00 (2006.01)

(21) International Application Number:
PCT/US2007/005891

(22) International Filing Date: 7 March 2007 (07.03.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/373,733 10 March 2006 (10.03.2006) US

(71) Applicant *(for all designated States except US)*: **UNISYS CORPORATION** [US/US]; Unisys Way, MS/E8-114, Blue Bell, PA 19424-001 (US).

(72) Inventor: **MAZZASATTI, Jane, Campbell**; 1798 Forest Creek Drive, Blue Bell, PA 19424 (US).

(74) Agents: **STARR, Mark, T.** et al.; Unisys Corporation, Unisys Way, MS/E8-114, Blue Bell, PA 19424-0001 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 2007/106365 A2

(54) Title: METHOD FOR PROCESSING AN INPUT PARTICLE STREAM FOR CREATING UPPER LEVELS OF KSTORE

(57) Abstract: A method for completing an incomplete sequence in a KStore having a particle stream, the particle stream having a plurality of input particles including at least one delimiter includes receiving the at least one delimiter within the particle stream to provide a received delimiter and first determining a current K node in accordance with the received delimiter. A match is second determined in accordance with the received delimiter and the current K node to provide a match determination. The KStore is provided with a list of defined delimiters and the second determining includes accessing the list. A determination is made whether the input particle is on the list. The current K node has an adjacent K node and the second determining includes locating the adjacent node in accordance with an asCase list of the current K node to provide a located asCase node.

METHOD FOR PROCESSING AN INPUT PARTICLE STREAM FOR CREATING UPPER LEVELS OF KSTORE

BACKGROUND OF THE INVENTION

1. FIELD OF INVENTION.

[0001] This invention relates to computing and, in particular to the field of database storage technology and the field of interlocking trees data stores.

2. DESCRIPTION OF RELATED ART

[0002] While interlocking trees datastores are covered in other patents by inventor Mazzagatti, it may be useful to provide a brief background summary of KStore and various features of said interlocking trees datastores.

[0003] A system and various methods for creating and using interlocking trees datastores and various features of the interlocking trees datastores have been developed. We refer to an instantiation of these interlocking trees datastores that we have developed as a KStore or just K. In particular, these structures and methods have been described in U.S. Patent 6,961,733 and copending patent application 10/666,382, (now published as 20050076011A1) by inventor Mazzagatti. Additionally, we described a system in which such interlocking trees datastores could more effectively be used in U.S. Serial No. 11/185,620, entitled "Method for Processing New Sequences Being Recorded into an Interlocking Trees Datastore." This invention provides the process invented to build and access the structure.

[0004] In U. S. Patent 6,961,733 and U.S. Serial Number 10/666,382, (now published as 50050076011), also by inventor Mazzagatti, we explained some preferred methods used to build and access an interlocking trees datastore. The methods taught in both of these patents were written at a level that taught the methodology of how an interlocking trees datastore is built and accessed.

[0005] All references cited herein are incorporated herein by reference in their entireties.

BRIEF SUMMARY OF THE INVENTION

[0006] A method for completing an incomplete sequence, or thought, in a KStore having a particle stream, the particle stream having a plurality of input particles including at least one delimiter includes receiving the at least one delimiter within the particle

stream to provide a received delimiter and first determining a current K node in accordance with the received delimiter. A match is second determined in accordance with the received delimiter and the current K node to provide a match determination. The KStore is provided with a list of defined delimiters and the second determining includes accessing the list of defined delimiters. A determination is made whether the input particle is on the list of defined delimiters. The current K node has an adjacent K node that is adjacent to the current K node and the second determining includes locating the adjacent node in accordance with an asCase list of the current K node to provide a located asCase node. The asCase list includes a plurality of asCase nodes and a plurality of adjacent nodes is located in accordance with the asCase list. If the learn functionality of the KStore is disabled, no further operations may be performed in accordance with the received delimiter if no adjacent node of the plurality of adjacent nodes has a Result node that matches the input delimiter. If the learn functionality of the KStore is enabled, Result node of the located asCase node is determined to provide a determined Result node, the second determining may include comparing the determined Result node with the received delimiter and a new node may be created.

[0007] The process used to create and access a K structure herein utilizes a procedure, which is called the praxis procedure. The praxis procedure may receive individual particles of incoming data, determine the type of particle and, based on the sensors and delimiters, access and construct the multiple levels of an interlocking trees datastore.

[0008] The KEngine creates and accesses a K structure from a stream of particles. Some of the particles in the particle stream may be identified as delimiters. Delimiters may be indicators that a portion of the particle stream is a complete sequence, or thought. As an example, a white space between characters in printed text indicates that one word is ending and another is beginning. The KEngine is required to recognize the delimiters and create K structure to record the represented data. Furthermore, the KEngine is designed to recognize and process particles as either delimiters or sensors. If a particle cannot be identified as either a delimiter or a sensor it may be ignored as noise.

[0009] Sensor particles are processed by the KEngine as extensions of a current sequence of events. If there is structure that has previously recorded the sequence, the

K may be traversed to reposition the current K location pointer. If there is no previous structure recording the sequence, new K structure may be created to record the event.

[0010] While the KEngine is processing the particle stream some particles are recognized as ending a sequence and beginning a new sequence. For example, within the field record universe the particle stream is divided into fields and groups of fields are divided into records. A common method of identifying the end of one field and the beginning of the next is to insert a particle, such as a comma, into the stream to indicate the limits of the field and a different character, such as a semi-colon, to indicate the limits of a record.

[0011] When the KEngine recognizes a comma particle, an EOT node may be appended to the current K path being created at a first level above the sensors, thereby completing a field entry. A new path beginning with the BOT node may then be established as the current K path for a further field entry. Particle processing then continues.

[0012] When the KEngine recognizes a semicolon particle, an EOT node may be appended to the current K path being created at the level above the field variable level. This may complete a record entry. A new K path beginning with the BOT node may be established as the current path for a record entry. In addition, the K path at the field variable below the record level may be completed and particle processing continues.

BRIEF DESCRIPTION OF SEVERAL VIEWS OF THE DRAWINGS

[0013] The invention will be described in conjunction with the following drawings in which like reference numerals designate like elements and wherein:

[0014] Figure 1 shows a block diagram representation of the main components which may be used with the present invention.

[0015] Figure 2A is a graphical representation of an interlocking trees datastore showing a structure representing the words CATS ARE FURRY.

[0016] Figure 2B is a graphical representation of a portion of the interlocking trees

datastore of Figure 2A showing a structure representing the word CATS.

[0017] Figure 2C is a graphical representation of a portion of the interlocking trees datastore of Figure 2A showing a structure representing the word CATS.

[0018] Figure 3 is a flowchart representation of a praxis procedure, which is a process that may match incoming particles of data with lists of delimiters, sensory data, and unidentified particles.

[0019] Figure 4 is a flowchart representation of a procedure for building and accessing a K structure from individual incoming particles of sensed data.

[0020] Figure 5A is a flowchart representation of a procedure for processing a delimiter.

[0021] Figure 5B is a flowchart representation of a procedure for processing a delimiter indicating a complete level of a K structure.

[0022] Figure 5C is a flowchart representation of a procedure for processing a delimiter and creating and accessing upper level subcomponent nodes.

[0023] Figure 6A is a diagram of an exemplary particle stream in a field/record universe of textual data containing a record with three fields and exemplary delimiters that separate each.

[0024] Figure 6B shows a generalized particlized stream using pixels as the individual data particles and exemplary delimiters that separate each.

DETAILED DESCRIPTION OF THE INVENTION

[0025] Referring now to Figure 1, there is shown a block diagram representation 100 of a KStore environment in which the system and method of the present invention may be implemented. Within such a KStore environment, information may flow bi-directionally between the KStore 14 and the remainder of the system through the K Engine 11. The transmission of information to the K Engine 11 may be by way of a learn engine 6 and the data source 8. The transmission of information may be by way of an API utility 5 and the application 7 as also understood by those skilled in the art. Providing graphical user interfaces 13, 12 to data source 8 and the application 7 may

thus permit an interactive user to communicate with the system.

The KEngine

[0026] The K Engine 11 receives a particle from somewhere outside the K engine 11 and creates or accesses the K structure 14. The K structure 14 contains elemental nodes that represent recognized particles of data. Figure 2A is a graphical representation of an interlocking trees datastore having the K structure for representing CATS ARE FURRY. The graphical representation of Figure 2A is used throughout this patent as an exemplary K structure for illustrative purposes.

[0027] Also represented within the K structure are the relationships that exist between the nodes. Each node in the K structure that is constructed may be assigned an address in memory. Additionally, each node may contain two pointers, a Case pointer and a Result pointer. The case pointer and the Result pointer of a node point to the two nodes from which it is formed. Also contained in a K node may be pointers to two pointer arrays, the asCase and the asResult array. The asCase array may contain pointers to the nodes whose Case pointers point to the K node. The asResult array, which contains pointers to the nodes whose Result pointers point to the K node. How the individual K nodes within a structure are constructed and accessed is the subject of numerous references by Mazzagatti, including U. S. Patent 6,961,733.

Data Particles

[0028] As mentioned above, data passed from the learn engine 6, the utilities 4 or the API utilities 5 to the K Engine 11 are particlized. For example, each word in a sentence may be treated as an individual particle of data, or each letter in a word may be treated as an individual particle of data. For example, in a textual data stream containing the words CATS ARE FURRY, the individual word CATS may be a particle, which may be sensed by a word particle sensor. Additionally, the word ARE and the word FURRY are particles which may be sensed by word particle sensors.

[0029] Each character or letter in a word, such as CAT, may be considered to be a particle which may be sensed by a sensor, in this case a character particle sensor (i.e., C is a particle of CAT as is A and T). Each of these may be a particle of data in a field/record textual universe of data. By textual it is meant that data are made up of

alphanumeric characters (e.g. the letters A through Z), special characters (e.g. punctuation) and numeric data (e.g. numbers). The term field/record is a carry over from traditional database terminology, wherein a field represents the title of a column in a table and a record represents the rows within the table and contains the actual data.

[0030] However, textual data is not the only type of data that may be streamed by the learn engine 6, utility 4 or API utility 5 into the K Engine 11. Those skilled in the art will understand that any kind of data that may be digitized may be particlized and streamed into K. For example, if the data universe is image data, the particles that may be digitized may be pixels. If the data universe is auditory data, the particles may be digitized sound waves. If the data universe is pressure data, particles may be digitized pressure values. If the data universe is olfactory data, particles may be digitized chemical molecules representing odors.

[0031] In many of the explanations that follow, the examples use data from the field/record universe. This means that in the examples, it is assumed that the data which is learned or accessed within K may come from traditional tabular databases or other traditional data structures in the form of text, numbers and special characters arranged in fields within records. But, it should be remembered that any type of data from any source that may be digitized may be learned and accessed within a K and therefore could have been used in the examples that follow. Also, the K structure may contain more than two levels of structure. As well, in the following, a KStore node diagram, as shown in Figure 2A, is used to illustrate an interlocking trees datastore depicting the creation of the words +CATS, +ARE and +FURRY and the sentence CATS ARE FURRY.

Generating an Interlocking Trees Datastore (K) from Particlized Data

[0032] As taught in U.S. Patent 6,961,733 and illustrated in Figure 1 herein, an exemplary system 100 for generating the interlocking trees datastore 14 in one embodiment may include the K Engine 11. The K Engine 11 may receive particles of data from a data stream from the learn engine 6, from the API utility 5 or from any other utility 4. The K Engine 11 is designed to recognize and process particles of data that it receives. Note that some of the particles may be created and used strictly within the K Engine 11. For example, BOT, end of list (EOL), end of record (EOR) or end of identity

(EOI) may be elemental nodes. In the current embodiment there are three types of particles that the K Engine may recognize: sensors, delimiters, and unidentified particles.

Praxis Procedure

[0033] A procedure that may recognize particles of sensor data, delimiters or unidentified particles according to the system and method of the invention may be the praxis procedure. Figure 3 shows a flowchart representation of a portion of the praxis procedure 300 which may be used for recognizing input particles in the system of the present invention. In the current embodiment, there may be three procedures corresponding to the three types of particles that may be received as input during the praxis procedure 300: (1) a procedure for processing a delimiter 301, (2) a procedure for processing unidentified particles (ignore sensor) 302 and (3) a procedure for processing sensor data 303. The following teaches the praxis procedure 300 in a preferred embodiment with special emphasis on how delimiters are processed and used to build and access an interlocking trees datastore consisting of multiple levels of K structure and how K location pointers or state are utilized.

Sensor Data, Delimiters, and Unidentified Particles

[0034] Before teaching in detail how sensor data, delimiters and unidentified particles are processed, it is necessary to explain what each of the three types of particles includes.

[0035] Sensor Data

[0036] A sensor may be any digitized data. A sensor is maintained within the K structure as an elemental root node. The elemental root nodes representing sensors may contain or point to values that match the digitized value of the sensor. In a field/record data universe, sensor data may include, but is not limited to, alphanumeric characters. The alphanumeric characters may include the letters in the alphabet, numbers and special characters such as punctuation and other special characters. Depending on how a system is configured a particle of sensor data may include only single letters, numbers, or characters, or they may be whole words, phrases, sentences, paragraphs, chapters, or even entire books, etc. Furthermore, particles may include

pixel values forming images of single letters or images of any other type. Thus, as mentioned above, data particles are not limited to textual data and may consist of any other forms of digitized data (e.g. pixels forming other images, sound waves, etc.).

Delimiters

[0037] Delimiters are particles that are used to identify an ending of a set of sensors. Furthermore, delimiters may be used to group sensor sets into hierarchies. For instance in a field/record universe, sets of letters may be grouped into words by delimiters. The words may then be grouped into field names or field values by delimiters. The field names or field values may be further grouped into fields and then into records.

[0038] Delimiters may be equivalent to individual sensors or sets of sensors. Or they may contain different values altogether. In the current embodiment, delimiters may include alphanumeric characters such as the letters of the alphabet, special characters such as, but not limited to, commas (,), semicolons (;), periods (.), and blanks ( ). Numbers in any base systems may also be used as delimiters. For example, in the current embodiment hexadecimal (base 16) numbers may be used as delimiters. However, as mentioned above, because particles are not limited to characters in the textual field/record universe, delimiters may also be any different type of digitized particle. For example, in a universe of digitized pixels, a single pixel or group of pixels may be used as a delimiter.

Unidentified Particles

[0039] Unidentified particles are any particles other than the ones that a current set of particle sensors and delimiter sensors recognizes. Unidentified particles, often called noise, may be, for example, particles of data from a different data character set (e.g. an Arabic or Chinese character). They may be particles from a different data universe, or they may just be an unprintable character that is not in the current set of sensors or delimiters.

Determining Particle Types

[0040] Refer back to Figure 3. As taught above, the praxis procedure 300 may

determine the particle type of an incoming particle received by a K Engine within a K system such as the K system 100. Based on the type of particle determined, the praxis procedure 300 may initiate one of three processes to process delimiters, sensor data or unidentified particles.

Comparing Particles to Delimiter List

[0041] In the praxis procedure 300 a particle of incoming data may be compared to a currently defined list of delimiters as shown in block 304. If the input particle matches an entry in the currently defined list of delimiters a process delimiter procedure is performed as shown in block 301. A process delimiter procedure that may be performed when a particle is determined to be a delimiter according to block 301 is taught below as the process delimiter procedure 500 in Figure 5A.

[0042] Comparing Particles to Sensor List

[0043] If the input particle does not match any of the current delimiters as determined according to the comparison of block 304 the praxis procedure 300 may continue to block 305. At block 305 the praxis procedure 300 may compare the incoming particle to a currently defined list of sensors.

[0044] The example in the following discussion uses the letter C as an exemplary particle of data from a textual field/record universe. Assume that in the example the letter C does not match any delimiter in the current set of delimiters and execution of the praxis procedure 300 proceeds to block 305. The praxis procedure 300 may then attempt to match the particle C with a list of current sensors in block 305. As taught in the above mentioned patents, in the current embodiment sensors may be maintained in the K structure as elemental root nodes. Lists of these elemental root nodes may be stored in arrays, hash tables, within the K 14 or a separate K structure or in any other manner understood in those skilled in the art.

[0045] For example, refer back to the exemplary structure shown in Figure 2A, which is a graphical representation of an exemplary interlocking trees datastore. The exemplary interlocking trees datastore includes structure representing the exemplary record CATS ARE FURRY. In this example, a particle C is found, for example, in a sensor array (not shown). Since there is a match, the praxis procedure 300 saves the

location of the elemental root node for the C particle to a variable to be used later. In this example, the location which is saved is location 225, as shown in Figure 2A.

[0046] It should be mentioned here that if the particle does not match anything in the sensor list, the ignore sensor process may be performed as shown in block 302 of Figure 3. The ignore sensor process may choose to discard any particle that is not recognized as a current sensor or delimiter, thereby treating it as noise. One skilled in the art will recognize that these discarded particles may be handled in numerous ways including notifying users via error or log files where other processes may be performed or users may review the contents. If the incoming particle matches something on the sensor list, the procedure of process sensor data block 303 is initiated.

Processing Sensor Data

[0047] Refer to Figure 4, which is a flowchart representation of a process sensor data procedure 400 according to the present invention. The process sensor data procedure 400 is suitable for processing sensor data to build or access a K structure according to an incoming particle of sensory data. Initiation of the process sensor data procedure 400 may occur pursuant to execution of the process sensor data block 303 within the praxis procedure 300, when an input particle does not match any entries in the current set of delimiters but does match an entry in the current set of sensors.

[0048] As shown in block 401 of the process sensor data procedure 400, the current K node on the current level of the K structure is determined, wherein terms such as "current K node," "current K location" and "current K pointer" are understood to refer to the location of the last experience on a selected level. When block 401 is executed the incoming particle has just been matched with the root node corresponding to the incoming particle according to block 305 of the praxis procedure 300. Therefore, the current level is known to be the level above the elemental root nodes. Accordingly, the current K node of the level above the root nodes is determined in block 401.

[0049] In a preferred embodiment of the invention, a list or any other kind of structure, may be maintained to store state variables indicating the current K location corresponding to each level. For example, in the case of a multilevel K structure an array setting forth the correspondence between each level of the K structure and a

variable indicating the current node of the level may be provided. The current K locations, or the current K node state data, of the levels of the K are known and stored according to the last event experienced on each level. The array or other data structure storing the current K node state data may be referred to as a state array or state table.

[0050] In one preferred embodiment each K location pointer may be used to identify both the current K level and the position on the current K level where the last event was experienced. Additionally, the foregoing structure for storing the correspondence between each level of the K structure and its current K node location pointer may store a list of the current set of delimiters, wherein the delimiters are described above with respect to block 304 of the praxis procedure 300 and in further detail below. However, the delimiter level data may be stored in any manner known to those skilled in the art. The structure may also contain a set of sensors appropriate for that particular level. The array of other data structure storing the current K state may be referred to as the state array or state table.

[0051] Furthermore, a correspondence between the defined delimiters and the levels of the K structure may be stored. Storage of this information permits the system to determine a relationship between an input delimiter and a level of the K structure that is being ended by the delimiter. It will be understood that the current K node state data and the delimiter level information do not need to be stored in the same data structure. It will also be understood that multiple delimiters may be appropriate for a single level.

[0052] As shown in block 402, the process sensor data procedure 400 may then determine the adjacent nodes of the current K node that was determined in block 401. As well known to those skilled in the art, the adjacent nodes of the current K node are determined by accessing an asCase list pointed to by an asCase pointer of the current K node. The asCase list contains pointers to each of the asCase nodes to be located in block 402. It will be understood by those skilled in the art that the asCase nodes located in this manner contain pointers to their Result nodes.

[0053] As shown in block 403, the Result nodes of the asCase nodes found in block 402 are determined according to their Result pointers. As shown in block 404, the Result nodes located in block 403 are then compared with the root node representing the received particle. If a match is found in decision 405 between a Result node of an

asCase node found in block 402 and an elemental root node representing an input particle, the matched asCase node becomes the current K node. Therefore, the first level K pointer is advanced to point to the matched asCase node as shown in block 407.

[0054] For example, assume that the current K node determined in block 401 is the beginning of thought (BOT) node 200 in Figure 2A. As described in block 402, the process sensor data procedure 400 determines the asCase nodes of the BOT node 200. In order to do this the asCase list of the BOT node 200 is examined. The nodes in the asCase list of the BOT node 200 are the nodes 205, 210, 215 and 220. It will thus be understood by those skilled in the art that each asCase node 205, 210, 215 and 220 includes a Case pointer pointing to the BOT node 200.

[0055] It will also be understood that each asCase node 205, 210, 215 and 220 includes a Result pointer pointing to its Result node. Thus, in block 403 the process sensor data procedure 400 may determine the Result node of each node 205, 210, 215 and 220 on the asCase list of the current K node by following its respective Result pointer to its respective root node. The Result nodes determined in this manner in block 403 may be compared with the elemental root node of the sensor corresponding to the received particle as shown in block 404. A determination may thus be made whether the Result node of any of the nodes 205, 210, 215 and 220 on the asCase list of the current K node match the elemental root node for the sensor of an input particle in block 404 of the process sensor procedure 400. The determination whether there is a match with the elemental root node for the sensor of the input particle may be made in decision 405.

[0056] Further to the foregoing example, the input particle in Figure 2A may be the letter particle C and the root node 225 may correspond to the value C of the input particle. If the Result nodes of the asCase nodes 210, 215, and 220 are compared in block 404 with the root node 225 no matches are found in decision 405 because none of the asCase nodes 210, 215 and 220 has a Result pointer pointing to the C elemental root node 225.

[0057] However, the asCase node 205 does contain a Result pointer pointing to the C elemental root node 225. Decision 405 of the process sensor data procedure 400

may therefore find that the Result node of the subcomponent node 205 is a match with the input particle. The current K location pointer may be set to the node +C 205, which has become the current K location of the level as shown in block 407. (For exemplary purposes in the diagrams, when the prefix notation "+" is placed before a value in a node in the figure, it indicates that the prefixed node has a valence, which will be understood to stand in for the entire thought up to but not including the prefixed node.) It will be understood that the asCase nodes of the current K node may be compared in any order and that once a match is found no more comparisons are needed.

[0058] In a different example, the current K location could be the subcomponent node 205 and the input particle could be the letter particle A. Pursuant to block 402 the asCase node of the node 205 is determined to be the subcomponent node 206. Since the Result node of the node 206 is the elemental root node representing the letter particle A, a match is found in decision 405. Thus, in block 407 the current K node is incremented to the subcomponent node 206.

[0059] Creating New Nodes

[0060] In some cases it may turn out that none of the nodes on the asCase list determined in block 402 has a Result pointer pointing to the root node of the input particle. Under these circumstances a match is not found in decision 405. Thus, it may be necessary to create new K structure as shown at block 408. The process of creating a new node is disclosed in several of the references incorporate herein, such as U. S. Patent No. 6,961,733 and U.S. Patent Serial Number 11/185,620, entitled "Method for Processing New Sequences Being Recorded Into an Interlocking Trees Datastore" for detailed explanation of how new nodes are created. Regardless of whether execution of the process sensor data procedure 400 proceeds by way of block 407 or by way of block 408 the intensity count may be incremented as shown in block 409.

Processing Delimiters

[0061] Refer back to Figure 3, showing the praxis procedure 300. As described in the foregoing description of the process sensor data procedure 400 of Figure 4, when a sensor is detected by the praxis procedure 300, execution of the praxis procedure 300 may proceed by way of block 303 to process the detected sensor in the process sensor

data procedure 400. However, the praxis procedure 300 may detect a delimiter particle rather than a sensor particle in an input particle stream. Under these circumstances the system and method of the invention may execute procedures suitable for processing the received delimiter.

[0062] As previously described, after comparing an input particle of data to the current list of delimiters in block 304 of the praxis procedure 300 a decision is made in decision 308 whether there is a match. If the input particle is found to match a currently defined delimiter in decision 308 the procedure of block 301 is initiated in order process the received delimiter. The procedure initiated by block 301 is the process delimiter procedure 500 of Figure 5A. Before teaching the process delimiter procedure 500 in detail, it is important to understand what delimiters are used for in the preferred embodiment of the invention.

[0063] In the preferred embodiment of the invention delimiters are used to indicate the end of a set of particle sequences of data as they are streamed into the K Engine 11. For example, as mentioned above, in the field/record universe, data may come from traditional databases in the format of fields and records.

[0064] Refer to Figure 6A showing a diagram of an exemplary particle stream 600. The exemplary particle stream 600 may represent a data record that may be stored in the K structure 14 and may therefore be referred to as the exemplary record 600. The exemplary particle stream 600 may represent three fields: Last Name 601, First Name 602, and Telephone Number 603. However, any number of fields of any size can be represented in other field/record universe particle streams, of which the exemplary particle stream 600 is but one example.

[0065] The first field in the exemplary particle stream 600 is the Last Name field 601 and is shown with the data sequence Cummings. The second field is the First Name field 602 and is shown with the data sequence William. The third field is the Telephone Number field 603 and is shown with the data sequence 7547860. At the end of the fields 601, 602 there is shown an end of field (EOF) delimiter 1D 604.

[0066] The hexadecimal character 1D 604 is thus used as an end of field delimiter for ending the first two fields 601, 602. However, the hexadecimal character 1E 605 is

used as both an end of field delimiter for ending the last field 603, and an end of record delimiter for ending the exemplary record 600. As such, it is a single delimiter that ends both the field 603 and exemplary particle stream 600, and, in general, in particle streams such as the exemplary particle stream 600 a delimiter is not required for closing each level of the KStore.

[0067] Thus, significantly, the hexadecimal character 1E 605 may be used to simultaneously end both: (i) its own level in the K structure (the record level), and (ii) a lower level of the K structure (the field level). Accordingly, in the embodiment of the invention represented by the exemplary particle stream 600, each level of a particle stream is not required to have its own separate closing delimiter. Furthermore, a higher level delimiter such as the delimiter 1E may complete any number of incomplete sequences, and thereby close any number of lower levels, in the manner that the field level of the exemplary particle stream 600 is closed.

[0068] Since textual data is not the only data that can be particlized and streamed into the K Engine 11, a more generalized explanation of delimiters may be helpful. In general, particles coming into the K Engine 11 may be thought of as incomplete sequences which can operate cooperatively to form complete sequences. Each incomplete sequence can represent an individual particle, set of particles of data, or the absence of particles. Individual incomplete sequences may be streamed into the K Engine 11 to form complete sequences. This is analogous to individual fields (incomplete sequences) such as the fields 601, 602, 603 forming a complete record (complete sequence) such as the complete record 600.

[0069] Figure 6B shows a more generalized stream of particles with incomplete sequences 606 making up a complete sequence 610. In Figure 6B each incomplete sequence 606 is shown as groups of pixels. However, incomplete sequences 606 could easily have been shown with textual data or data from any other data universe. In the complete sequence 610 the EOT delimiter 607 is shown as the hexadecimal character 1D and the final end of product delimiter 608 is shown as the hexadecimal character 1E. This relationship is shown in Figure 2A at the nodes 265, 282.

[0070] Although the hexadecimal characters 1D and 1E are used as delimiters 607,

608 in the illustrative examples, it will be understood that any other particle may be defined to serve as delimiters 607, 608. For example, a comma, another numerical character including characters that are not hexadecimal characters or a specific group of pixels. Thus, delimiters may be any particle that is defined as such for the praxis procedure 300 when the processing of the delimiter particles begins.

[0071] It should be noted that incomplete sequences are not limited to single particles of data. An incomplete sequence may be any sequence of data that is experienced before an EOT delimiter is experienced. An incomplete sequence may also include the absence of particles indicating a null value, terminated by an EOT delimiter.

[0072] Again referring back to the praxis procedure 300 in Figure 3, an incoming particle may be compared to a list of currently defined delimiters as shown in block 304. If the input particle matches one of the currently defined delimiters as determined in decision 308, the procedure of process delimiter block 301 can be initiated to process the received delimiter particle. The procedure for processing the received delimiter particle according to process delimiter block 301 is the process delimiter procedure 500 of Figure 5A.

[0073] Refer now to Figure 5A, which is a flowchart representation of the process delimiter procedure 500 for processing delimiters found in an input particle stream. The process delimiter procedure 500 can be initiated by the process delimiter block 301 of the praxis procedure 300 when a match is found between an input particle and an entry on the list of currently defined delimiters by decision 308.

[0074] As previously described, it is possible for the praxis procedure 300 to receive a higher level delimiter for completing its own level of the K structure while lower levels of K structure are still incomplete. Under these circumstances, the higher level delimiter may complete as many incomplete lower levels as necessary prior to completing its own level.

[0075] For example, refer above to the exemplary particle stream 600 shown in Figure 6A. An EOF delimiter hexadecimal 1D 604 is shown at the ends of the fields 601, 602. The hexadecimal delimiter character 1D 604 is thus used as the delimiter for

the first two fields 601, 602. However, there is no delimiter character 1D 604 at the end of the field 603. Rather, only the hexadecimal delimiter character 1E 605 is shown at the end of the field 603, wherein it is understood that the level of the delimiter character 1E 605 is higher than the level of the field 603. Therefore, the received delimiter character 1E 605 is used to indicate both the end of the last field 603, and the end of the exemplary particle stream 600. Under these circumstances, the received delimiter character 605 performs both the operation of completing the incomplete sequence 603, at a lower level, and the operation of ending the record 600, at a higher level.

[0076] Thus, at the time the delimiter character 605 is received: (i) the field 603 represents an incomplete sequence on an incomplete lower level, and (ii) the delimiter character 605 is a delimiter for a higher level of K structure than the current level of field 603. Accordingly, the system and method of the present invention may determine both: (i) that the level of the field 603 must be completed, and (ii) that the level of the record 600 must be completed. Additionally, the system and method of the present invention may perform the operations necessary for completing both the field 603 and the record 600.

[0077] Furthermore, those skilled in the art will understand that a received delimiter may indicate the end of any number of lower levels in the manner that the delimiter character 605 indicates the end of only a single lower level. Accordingly, the system and method of the invention may perform the operations necessary for completing as many lower levels as required in addition to completing the level of the received delimiter.

[0078] Therefore, the process delimiter procedure 500 of Figure 5A is provided to perform the operations of completing as many incomplete levels as necessary below the level of a received delimiter, as well as completing the level of the received delimiter itself. In block 501 of the process delimiter procedure 500 the level associated with the input delimiter is determined. This determination may be made according to a list of currently defined delimiters and the K location structure or state structure setting forth the corresponding delimiter level as previously described. Additionally, the variable Input Delimiter Level is set equal to the determined level in block 501.

[0079] As previously described in the current embodiment, sets of particle

sequences, such as the sets of sequences forming the incomplete sequences 606 in Figure 6A, may be entered into the K structure 14 in levels. Thus, in effect, hierarchy is determined by the organization or location of the delimiters. For example, any number of levels may appear in a K structure and multiple types of end product nodes may be present in any one level. Refer back to Figure 2A. The interlocking trees datastore shown in Figure 2A includes three exemplary levels: 0, 1 and 2. An individual K structure is not limited to three levels and may contain as many as necessary. Note that the level numbers indicated in these descriptions are used for the sake of clarity of the discussion. Levels may be linked by any means desired with the concept of an "upper" level being relative to whatever linked structure is utilized. The structure used to link the levels, as discussed previously for the K location pointers or state structure, may be an array, a linked list, a K structure or any other structure known to those skilled in the art.

[0080] Level 0 (230) of the K shown in Figure 2A may represent the elemental root nodes. For example, using field/record textual universe data of Figure 2A, level 0 may represent the elemental root nodes 200, 225, 271, 265, or 282 as well as the other elemental root nodes that have not been provided with reference numerals in Figure 2A.

[0081] Level 1 (235) may represent the subcomponent nodes and end product nodes of the paths 240, 245 and 250. The Result pointers of the nodes in level 1 point to the elemental root nodes in level 0.

[0082] For example, the path 240 includes the nodes 200, 205, 206, 207, 208 and 260. Assume that a delimiter for end of field, such as the delimiter 1D 265 similar to the delimiter 1D 604 in figure 6A, is recognized while the K location pointer for level 1 is positioned at the exemplary node 208. The nodes of the path 240 from the BOT node 200 to the node 208 thus represent an incomplete sequence for the exemplary sequence BOT-C-A-T-S. The delimiter 1D 265 recognized at this point indicates the termination of the field sequence from the BOT node 200 to the node 208. Thus, an end product node 260 may be built. The addition of the end product node 260, having the EOT delimiter 1D 265 as its Result node, completes the incomplete sequence, and the exemplary word CATS is thus represented by the path 240. It is the recognition of a delimiter 1D in this manner, after experiencing an incomplete sequence, that completes the sequence.

[0083] Level 2 (255) represents the subcomponent nodes whose Result pointers point to the complete sequences of level 1 in Figure 2A. The complete sequences of level 1 are represented by the end product nodes +CATS 260, +ARE 270 and +FURRY 275. The addition of the end product node 283, having the EOT delimiter 1E 282 as its Result node, may be used to complete the incomplete sequence, thus completing the record CATS ARE FURRY.

[0084] Referring back to Figure 5A. As explained above, in block 501 of the process delimiter procedure 500 an incoming delimiter is associated with its defined level within the interlocking trees datastore and the variable Input Delimiter Level is set equal to the associated level. For example, within a field/record universe the exemplary hexadecimal character 1D 607 in figure 6A may be used to represent the end of a field 606 (i.e. the end of a complete field sequence) as previously described. As also described, the exemplary hexadecimal character 1E may be used to represent the end of a record (i.e. the end of a complete record sequence). Both of the delimiters 1D, 1E in the current embodiment may initiate processing that indicates completion of a specific level within the K structure. Thus, the level is identified with which the experienced delimiter is associated.

[0085] The process delimiter procedure 500 may next determine which, if any, levels lower than Input Delimiter Level are incomplete at the time the input delimiter is received. This determination may be made with reference to the list of the current K nodes in the K structure. As previously described, this list may contain the current K pointers for each level of the K structure. In one embodiment the K location pointer for each level may indicate the node in that level where the last event for that level was experienced, and the K location pointer for completed levels can point to any location designated as a sequence beginning location. In one preferred embodiment the sequence beginning location can be the BOT node 200. The process for ending the incomplete sequences located in this manner may begin with the lowest such level as shown in block 502. The lowest such level, in general, can be any level of the KStore. Execution of the process delimiter procedure 500 may then proceed to block 503 where the process complete level procedure 550 of Figure 5B is initiated in order to begin ending incomplete sequences as necessary.

[0086] For example, in Figure 2A, assume that a previous particle S 271 in the

sequence BOT-C-A-T-S was the last particle sensed in level 1 (235). The sensing of the particle S 271 may permit the forming of the incomplete sequence at the node 208, as previously described. At this point, the K location pointer for level 1 points to the node 208, thereby indicating that the last event experienced on level 1 (235) was at the node 208. Thus, level 1 is incomplete at this point. Therefore, level 1 is the starting level determined in block 502 of the process delimiter procedure 500 when a delimiter 1D is received. The incomplete sequence +S 208 may be completed by the process complete level block 503 which initiates the process complete level procedure 550 of Figure 5B.

[0087] Refer to Figure 5B, which shows the process complete level procedure 550. In a preferred embodiment of the invention, the process complete level procedure 550 is initiated by the execution of block 503 of the process delimiter procedure 500 when an incomplete level is determined. The process complete level procedure 550 is adapted to complete the processing of the incomplete levels determined in block 502. The presence of unfinished lower level can be determined with reference to the table of current K node pointers of each level as previously described. The lower levels are closed starting from the lowest incomplete level and proceeding upward through the determined level.

[0088] In block 504 of Figure 5B, the Result nodes of the asCase nodes of the current K node are compared with the determined delimiter. The process of block 504 is substantially similar to the operations of blocks 401-404 of the process sensor data procedure 400 described above. In decision 505 a decision is made whether any of the asCase nodes of the current K location for the determined current K level have a Result node that matches the root node for the determined delimiter. If no matches are found in decision 505 an end product node has not been built and processing continues to block 506. In block 506 a new end product node can be created in order to complete the incomplete sequence of the determined current K level and the current K location pointer is set to the new node.

[0089] Refer to Figure 2B, which illustrates a K structure in the process of being built. In this exemplary figure, assume again that the node 208 is the last node formed and that the input particle received matched the level 1 delimiter 1D. Therefore, the K location pointer for level 1 points to the node 208. As explained above, the asCase list

of the current K node 208 is checked. It is determined by decision 505 that there are no nodes in the asCase list of node 208. Therefore, processing of the process complete level procedure 550 proceeds to block 506 where the end product node 260 is created. The end product node 260 created in this manner links the node 208 to the elemental root node 265 for the field delimiter 1D for the current level which in this case is level 1. The K location pointer for level 1 is then set to the node 260 where it indicates that the level is complete. In this exemplary figure, the end product node 260 is in level 1.

[0090] In a further example of the case in which execution of the process complete level procedure 550 proceeds from decision 505 and builds a new node, assume that the current K pointer is pointing to the subcomponent node 274 of Figure 2A when the delimiter 1D is received. If the +EOT node 275 has not previously been built the decision 505 of the process complete level procedure 550 will not find any asCase nodes. Under these circumstances processing may proceed to block 506 where the end product node 275 may be created, as described in the foregoing example.

[0091] However, when an end product asCase node of a current K node has already been experienced and built, execution of the process complete level procedure 550 may proceed from decision 505 to block 507. For example, if the field represented by the path 250 has previously been experienced by the K structure at least once, the asCase list of the node 274 is not empty. Thus, a comparison between the Result node of the asCase node 275 and the elemental root node for the delimiter may be positive. In the current example, such a match is found because the asCase node (the node 275) of the current K node (274) does, in fact, have a Result pointer pointing to the ID delimiter sensor 265.

[0092] Thus, in this example, execution of the process complete level procedure 550 may proceed to block 507. In block 507 the previously existing node 275 may become the current K node and the count of the nodes may be incremented.

[0093] Whether execution of the process complete level procedure 550 proceeds by way of block 506 to create a new node and advance the current K pointer, or by way of block 507 to merely advance the current K pointer to a preexisting node, the count of the node is incremented and a determination is made whether there are potentially any higher levels above the current level as shown in decision 508. The determination

whether there are higher levels is made by accessing the list of defined delimiters as previously described and determining where the determined delimiter is located in the defined hierarchy.

[0094] If there are no levels higher than the current K level, the K location pointer is set to the BOT node 200 to indicate that the current K level is complete as shown in block 509. The system may then wait for the next input particle. Processing by the process complete level procedure 550 is then complete. Processing may then return to the process delimiter procedure 500 in Figure 5A and proceed from block 503 to block 511. If there is a higher level in the K structure, as determined in block 508, processing continues to the process upper level subcomponent block 510 where a subcomponent node may be built if necessary. The processing performed by the process upper level subcomponent block 510 initiates the process upper level subcomponent procedure 590 shown in Figure 5C.

[0095] Refer to Figure 5C, which is a flowchart representation of the process upper level subcomponent procedure 590. The process upper level subcomponent procedure 590 is initiated by process upper level subcomponent node block 510 of the process complete level procedure 500.

[0096] The upper level subcomponent procedure 590 may begin with blocks 514a-d. The operations of blocks 514a-d of the process upper level subcomponent procedure 590 are substantially similar to the operations of blocks 401-404 of the process sensor data procedure 400 described above

[0097] As shown in block 514a, the current K node on the upper level may be determined. For example, referring back to Figure 2B, the current K node on the upper level (255) may be the BOT node 200. As shown in block 514b, the asCase list of the BOT node 200 may be used to locate the asCase nodes of the BOT node 200. The node 205 is thus located. As shown in block 514c, the Result pointers of the asCase nodes of the BOT node 200 are followed to find any Result nodes. The elemental root node 225 is thus located. As shown in block 514d, the Result node located in this manner is compared with the end product node for the previous level node 260.

[0098] In decision 515 a decision is made whether any of the asCase nodes of the

current K location for the current level have a Result node that matches the root node or end product node for the previous level. If there is a match the upper level K location pointer is set to the matched node as shown in block 516. However, if the end product node has not been experienced before at this level then no matches are found by decision 515 and processing continues to block 517. In block 517 a new subcomponent node may be created in the higher level and the current K location pointer for the higher level may be set to the new node.

[0099] For example, refer to Figure 2C, which is a graphical representation of a portion of an interlocking trees datastore, for example, a portion of the interlocking trees datastore that was originally shown in Figure 2A. The datastore in Figure 2C was previously begun in Figure 2B, as previously described. However, the datastore of Figure 2C has an additional node, not present in the datastore of Figure 2B, the level 2 subcomponent node 220 representing the sequence BOT- CATS. The Result node of the node 220 is the +EOT node 260 of level 1. The +EOT node 260 is the end product node of the path 240 representing BOT-C-A-T-S-EOT.

[00100] Further to Figure 2B, the current K location for the upper level or level 2 (255), is the BOT node 200. At this point the asCase list of the BOT node 200 is checked and found to contain only one node, the node 205. The Result pointer for the node 205 is then checked and found to point to the elemental root node 225. The elemental root node 255 represents the particle C.

The elemental root node 205 thus does not match the end product node pointed to by the K location pointer for level 1, the +EOT node 260. Now refer to figure 2C. In Figure 2C, a new subcomponent node may be created at the upper level (255), which in this exemplary case is the BOT-CATS node 220. The subcomponent node 220 is then set as the current K location node for the upper level. Processing then returns to Figure 5B and proceeds from block 510 to block 509 where the current K location pointer for level 1 (235) is set to the node BOT 200. After completion of block 509 the K location pointer for level 1 points to the BOT node 200 and the K location pointer of level 2 points to the node 220. Processing may then continue to block 511 of Figure 5A by way of calling block 503. Processing Upper Levels

[00101] The foregoing descriptions disclose how delimiters may signal the end of

complete sequences at lower levels (e.g. field levels in a field/record data universe). The following discussion discloses how delimiters are used to signal the end of complete sequences at upper levels (e.g. record levels in a field/record data universe). In this part of the explanation, assume that portions of an upper level have already been established.

[00102] It will be understood that to some extent the procedures for completing upper levels are similar to those for completing the lower levels as they were previously described. Therefore, where the following procedures are similar to those that have previously been taught above, the explanation may refer back to the earlier explanations. Also, the following discussion is taught using the exemplary delimiters from the field/record universe. And, before continuing, some assumptions may be made before explaining in detail how the upper level delimiters are processed.

Process Upper Level When Lower Levels are Complete

[00103] Assume in the following discussion that a K structure such as K 14 shown in Figure 2A continues to be built. Also assume that the lower level delimiters (e.g. the 1D delimiter in the exemplary case) are experienced at the end of incomplete sequences, thereby completing the incomplete sequences. Also assume that eventually an upper level delimiter, e.g. 1E in a field/record universe, is experienced. Again, it should be noted that particles from a field/record universe are not the only particles that the K Engine 11 may process. Additionally, the delimiters used in the following examples (hexadecimal characters 1D and 1E) are not the only delimiters that may be used within the KStore system. Furthermore, those skilled in the art will realize that the praxis procedure 300 of the invention is not limited to field/record data, and that any data that can be digitized (e.g. pixels) may be represented as a K structure through the praxis procedure 300.

[00104] As mentioned above, the following discussion uses the K structure shown in Figure 2A to explain the process of completing the upper levels of a K structure. As the following discussion begins, refer to Figure 2A and assume the following about each level.

- Level 0 (230) - Contains all of the elemental root nodes of the K Store 14.

- Level 1 (235) – The paths 240, 245, and 250 are complete. The K location pointer for level 1 points to the BOT node 200.

- Level 2 (255) – The sequences that can be represented by the subcomponent nodes 220, 280, and 281 have been processed and the K location pointer for the level 2 points to the node 281.

[00105] As the following discussion begins, the next particle that is experienced is the delimiter 1E, wherein the delimiter 1E closes its own level (level 2) as shown in the exemplary particle string 610 of Figure 6A.

[00106] As explained above, the praxis process 300 shown in Figure 3 begins in block 304 by determining whether the received particle is a currently defined delimiter. Since the particle is a delimiter, execution proceeds to the process delimiter procedure 500 of Figure 5A by way of block 301 of Figure 3.

[00107] Refer back to the process delimiter procedure 500 in Figure 5A, which is a flowchart representation of a procedure for processing delimiters. Since in the example the received hexadecimal character 1E is defined to represent an end of record, it is known that this delimiter is associated with level 2 (255) by accessing the delimiter level data or state structure as shown in block 501. The process shown in block 502 determines that the lowest incomplete level is level 2 (255) because the K location pointer for level 1 (235) is at BOT node 200.

[00108] Again, as explained above in detail, the process complete level procedure 550 shown in Figure 5B is initiated by way of block 503. The procedure steps shown in blocks 504, 505 and 506 are completed and the end product node +EOT 283 is created in block 506 and set as the K location pointer for level 2. When the procedure 550 reaches block 508, a determination is made whether there are any potentially higher levels within the KStore. In the exemplary case, no other higher level delimiters are defined beyond the hexadecimal character 1E. Thus, there are no other higher levels in the K. Therefore, the K location pointer for level 2 (255) is set to the BOT node 200 as shown in Figure 2A and block 509 of Figure 5B.

[00109] From block 509, the process complete level procedure 550 returns to the calling block 510 in Figure 5A and proceeds to block 511. In block 511 the level is set

to the next upper level. Since there is no level higher than this one, the current level is set to a value larger than the maximum level, in this case level 3. In blocks 512 the current level is compared to the Input Delimiter Level and in block 513 of the procedure 500 determines whether the current level is greater than the level of the input delimiter. In the example, the input delimiter is at level 2. Since level 3 is greater than level 2, the question in decision block 513 is answered YES, indicating completion of the delimiter processing in the procedure 500. Execution may then return to block 303 of the praxis procedure 300 in Figure 3. At this point the praxis procedure 300 may return to its calling procedure, block 301, where the system awaits the next incoming particle.

Process Upper Level When Lower Levels are not Complete

[00110] Assume in the following discussion that a K structure such as K 14 shown in Figure 2A continues to be built. Also assume that the last lower level delimiter (e.g. the 1D delimiter in the exemplary case) has not yet been experienced at the end of the last incomplete sequence. Also assume that eventually an upper level delimiter, e.g. 1E in a field/record universe, is experienced. Again, it should be noted that particles from a field/record universe are not the only particles that the K Engine 11 may process. Additionally, the delimiters used in the following examples (hexadecimal characters 1D and 1E) are not the only delimiters that may be used within the KStore system. Furthermore, those skilled in the art will realize that the praxis procedure 300 of the invention is not limited to field/record data, and that any data that can be digitized (e.g. pixels) may be represented as a K structure through the praxis procedure 300.

[00111] As mentioned above, the following discussion uses the K structure shown in Figure 2A to explain the process of completing the upper levels of a K structure. As the following discussion begins, refer to Figure 2A and assume the following about each level.

- Level 0 (230) - Contains all of the elemental root nodes of the KStore 14.
- Level 1 (235) – The paths 240 and 245 are complete. Within the path 250, the sequences that may be represented by the nodes 215, 216, 272, 273 and 274 have been experienced, and the K location pointer for level 1 points to the node 274.
- Level 2 (255) – The sequences that may be represented by the subcomponent

nodes 220 and 280 have been processed and the K location pointer for the level 2 points to the node 280.

[00112] As the following discussion begins, the next particle that is experienced is the delimiter 1E, wherein the delimiter 1E closes both its own level (level 2) and the level below it (level 1) as shown in the exemplary particle string 600 of Figure 6A. Thus, in general, in particle streams such as the exemplary particle stream 600 a delimiter is not required for closing each level of the KStore.

[00113] As explained above, the praxis process 300 shown in Figure 3 begins in block 304 by determining whether the received particle is a currently defined delimiter. Since the particle is a delimiter, execution proceeds to the process delimiter procedure 500 of Figure 5A by way of block 301 of Figure 3.

[00114] Refer back to the process delimiter procedure 500 in Figure 5A, which is a flowchart representation of a procedure for processing delimiters. Since in the example the received hexadecimal character 1E is defined to represent an end of record, it is known that this delimiter is associated with level 2 (255) by accessing the delimiter level data or state structure as previously described. The process shown in block 502 determines that the lowest incomplete level is level 1 (235) because the K location pointer for level 1 (235) is not at BOT node 200. Rather, it points to the subcomponent node 274 of the K path 250 within level 1 (235) in the current example. It is also determined from the delimiter level data or state structure that the delimiter for level 1 is 1D.

[00115] As explained above, the process delimiter procedure 500 may proceed by way of block 503 to initiate the process complete level procedure 550 of Figure 5B, in order to complete the incomplete lower level 1 (235) of the K before processing the upper level (255). The level, level 1, and the determined delimiter, 1D, are passed to the process complete level procedure. In block 504 the asCase node of the K location pointer for this level (level 1), node 274, if any, is located. If the +EOT node 275 has already been created there is a match in decision 505 between its Result node 265 and the determined delimiter, wherein it is understood that the determined delimiter 1D is the delimiter associated with level 1 (235). The current K node for level 1 is advanced to point to the +EOT node 275 in block 507 and the intensity is incremented.

[00116] If the +EOT node 275 has not already been created, there is no end product node and no match in decision 505. The process complete level procedure 550 may then proceed to block 506 where the +EOT node 275 may be created. Since the new node is to be located on level 1(235) the Result node of the new +EOT node 275 is set to EOT 1D 265.

[00117] The procedure 550 may increment the count and proceed to decision 508 where a determination may be made whether there are any higher levels. Because there is a level above level 1 (235), namely level 2 (255), the process upper level subcomponent procedure 590 of Figure 5C is initiated by way of block 510.

[00118] As the process upper level subcomponent procedure 590 of Figure 5C is initiated by way of block 510 of Figure 5B, the procedures in blocks 514a-d are performed. In these operations the asCase nodes, if any, of the current K node (the node 280) of level 2 (255) may be located. The Result nodes of any asCase nodes located can be compared to the end product node for the previous level. In the current example the asCase node 281 may be located. The Result node of the asCase node 281 is compared with the end product or root node of the previous level or node 275. Since node 275 matches the K location pointer for the previous level, the K location pointer for the upper level or level 2 is set to node 281 representing "BOT-CATS-ARE-FURRY", as shown in Figure 2A. If there had been no match a new subcomponent node would have been created in block 517 and the current K location for level 2 advanced to the newly created node. The process returns to Figure 5B block 509, at which point the K location pointer for level1 is set to BOT. The process then returns to Figure 5A block 511.

[00119] The current level is then set to the next highest level in block 511 of the process delimiter procedure 500. In the current example the next highest level is delimiter level 2 (255). This is the record level in the field/record universe of data of the current example. As shown in block 512 of the process delimiter procedure 500 the new level is compared to the variable Input Delimiter Level of block 501. In the example, the input delimiter is 1E, which represents level 2 (235), and the current K level is also level 2 (235). In the decision block 513 a determination is made whether the current K level is greater than the variable Input Delimiter Level. Since both level numbers are 2 in the current example the answer to decision 513 is NO. The process

delimiter procedure 500 may therefore proceed from the decision 513 by way of the process complete level block 503 to the process complete level procedure 550 of Figure 5B to complete the processing for level 2 (255).

[00120] Again, as explained above in detail, the process complete level procedure 550 shown in Figure 5B is initiated. The procedure steps shown in blocks 504, 505 and 506 are completed and the end product node +EOT 283 is set as the K location pointer for level 2. When the procedure 550 reaches block 508, a determination is made whether there are any potentially higher levels within the KStore. In the exemplary case, no other higher level delimiters are defined beyond the hexadecimal character 1E. Thus, there are no other higher levels in the K. Therefore, the K location pointer for level 2 (255) is set to the BOT node 200 as shown in Figure 2A and block 509 of Figure 5B.

[00121] From block 509, the process complete level procedure 550 returns to the calling block 510 in Figure 5A and proceeds to block 511. In block 511 the level is set to the next upper level. Since there is no level higher than this one, the current level is set to a value larger than the maximum level or, in this case, level 3. In blocks 512 the current level is compared to the Input Delimiter Level and in block 513 of the procedure 500 determines whether the current level is greater than the level of the input delimiter. In the example, the input delimiter is at level 2. Since level 3 is greater than level 2, the question in decision block 513 is answered YES, indicating completion of the delimiter processing in the procedure 500. Execution may then return to block 303 of the praxis procedure 300 in Figure 3. At this point the praxis procedure 300 may return to its calling procedure, block 309, where the system may await the next incoming particle.

[00122] **Claims**

[00123] 1. A method for completing an incomplete sequence in a KStore having a particle stream, said particle stream having a plurality of input particles including at least one delimiter, comprising:

receiving said at least one delimiter within said particle stream to provide a received delimiter;

first determining a current K node in accordance with said received delimiter; and

second determining a match in accordance with said received delimiter and said current K node to provide a match determination.

[00124] 2. The method for completing an incomplete sequence in a in a KStore of claim 1, wherein said KStore is provided with a list of defined delimiters and said second determining comprises accessing said list of defined delimiters.

[00125] 3. The method for completing an incomplete sequence in a KStore of claim 2, further comprising determining whether said input particle is on said list of defined delimiters.

[00126] 4. The method for completing an incomplete sequence in a KStore of claim 1, wherein said current K node has an adjacent K node that is adjacent to said current K node and said second determining comprises locating said adjacent node in accordance with an asCase list of said current K node to provide a located asCase node.

[00127] 5. The method for completing an incomplete sequence in a KStore of claim 4, wherein said asCase list includes a plurality of asCase nodes further comprising locating a plurality of adjacent nodes in accordance with said asCase list.

[00128] 6. The method for completing an incomplete sequence in a KStore of claim 5, wherein a learn function of said KStore is disabled further comprising performing no further operations with said received delimiter if no adjacent node of said plurality of adjacent nodes has a Result node that matches said input delimiter.

[00129] 7 The method for completing an incomplete sequence in a KStore of claim 4, wherein said second determining further comprises determining a Result node of said located asCase node to provide a determined Result node.

[00130] 8. The method for completing an incomplete sequence in a KStore of claim 7, wherein said second determining further comprises comparing said determined Result node with said received delimiter.

[00131] 9. The method for completing an incomplete sequence in a KStore of claim 1, wherein said match determination is negative further comprising building a new asCase node of said current K node in accordance with said negative match determination.

[00132] 10. The method for completing an incomplete sequence in a KStore of claim 1, wherein said match determination is positive to provide a matched node further comprising setting said current K node to said matched node.

[00133] 11. The method for completing an incomplete sequence in a KStore of claim 9, wherein said new asCase node comprises an end product node.

[00134] 12. The method for completing an incomplete sequence in a KStore of claim 9, further comprising setting said new asCase node as a new current K node.

[00135] 13. The method for completing an incomplete sequence in a KStore of claim 9, further comprising incrementing a node count.

[00136] 14. The method for completing an incomplete sequence in a KStore of claim 1, wherein said match determination is positive further comprising setting said new asCase node as a new current K node.

[00137] 15. The method for completing an incomplete sequence in a KStore of claim 14, further comprising incrementing a node count.

[00138] 16. The method for completing an incomplete sequence in a KStore of claim 1, wherein said KStore includes a plurality of KStore levels having respective current K nodes.

[00139] 17. The method for completing an incomplete sequence in a KStore of claim 16, wherein said first determining comprises accessing current K node data associating said KStore levels of said plurality of KStore levels with their respective current K nodes.

[00140] 18. The method for completing an incomplete sequence in a KStore of claim 17, wherein said determining of said current K node further comprises determining a KStore level of said plurality of KStore levels in accordance with said received delimiter.

[00141] 19. The method for completing an incomplete sequence in a KStore of claim 1, further comprising providing delimiter level data.

[00142] 20. The method for completing an incomplete sequence in a KStore of claim 19, further comprising accessing said delimiter level data in accordance with said received delimiter.

[00143] 21. The method for completing an incomplete sequence in a KStore of claim 20, wherein said KStore includes a plurality of KStore levels, a plurality of delimiters and a state data structure for representing associations between said KStore levels and said delimiters further comprising determining a current K level in accordance with said state data structure.

[00144] 22. The method for completing an incomplete sequence in a KStore of claim 1, further comprising determining whether said KStore includes any KStore levels higher than said current KStore level to provide a higher KStore level determination.

[00145] 23. The method for completing an incomplete sequence in a KStore of claim 22, wherein said KStore is provided with current K node data further comprising determining a further node on a higher KStore level in accordance with said current K node data and setting said further node as a further current K node.

[00146] 24. The method for completing an incomplete sequence in a KStore of claim 23, wherein said further node comprises a subcomponent node.

[00147] 25. The method for completing an incomplete sequence in a KStore of claim 1, further comprising setting said current K node to a sequence beginning location.

[00148] 26. The method for completing an incomplete sequence in a KStore of claim

25, wherein said sequence beginning location comprises a BOT node.

[00149] 27. The method for completing an incomplete sequence in a KStore of claim 1, wherein said KStore includes a plurality of KStore levels further comprising:

determining a KStore level of said plurality of KStore levels in accordance with said received delimiter to provide a current KStore level; and

determining whether said KStore includes any KStore levels of said plurality of KStore levels higher than said current KStore level to provide a higher KStore level determination.

[00150] 28. The method for completing an incomplete sequence in a KStore of claim 27, wherein said further match determination is negative.

[00151] 29. The method for completing an incomplete sequence in a KStore of claim 28, further comprising building a new asCase node of said higher level current K node when said further match determination is negative.

[00152] 30. The method for completing an incomplete sequence in a KStore of **claim 29**, wherein said new asCase node comprises a subcomponent node.

[00153] 31. The method for completing an incomplete sequence in a KStore of **claim 30**, further comprising setting said subcomponent node as a further current K node.

[00154] 32. The method for completing an incomplete sequence in a KStore of claim 27, further comprising determining a Result node of an asCase node of a further current K node of said higher KStore level to provide a further determined Result node.

[00155] 33. The method for completing an incomplete sequence in a KStore of claim 32, further comprising comparing said further determined Result node with a determined end product node at said lower level to provide a matched node.

[00156] 34. The method for completing an incomplete sequence in a KStore of claim 32, wherein said match determination is positive further comprising setting said asCase node as a new upper level current K node.

[00157] 35. The method for completing an incomplete sequence in a KStore of claim

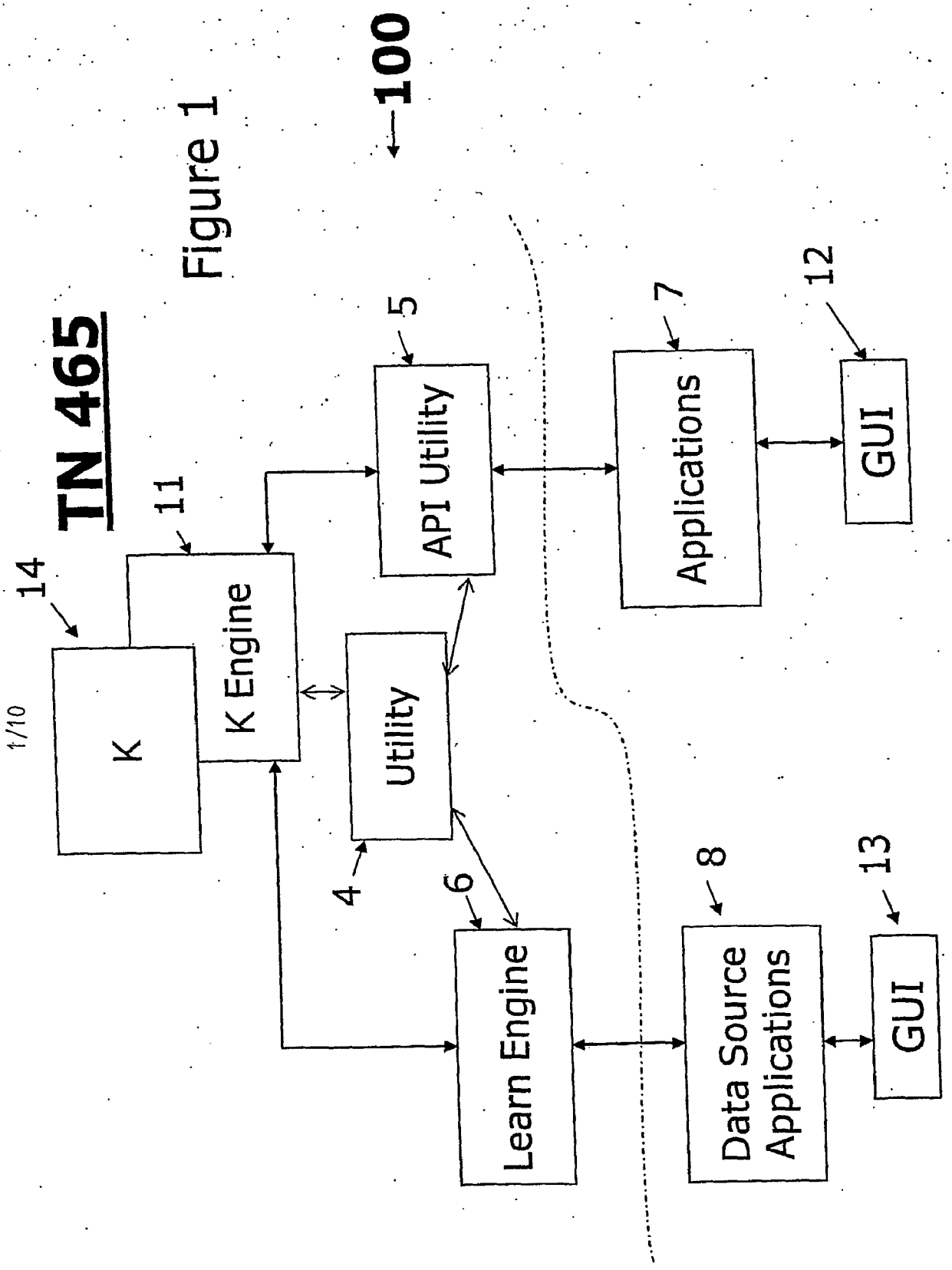
34, further comprising incrementing a node count.

[00158] 36. The method for completing an incomplete sequence in a KStore of claim 1, further comprising receiving no sensor data within said incomplete sequence prior to receiving said at least one delimiter.

[00159] 37. The method for completing an incomplete sequence in a KStore of claim 27, wherein said match determination is negative and a learn function of said KStore is disabled further comprising performing no further operations with said received delimiter.

[00160] 38. The method for completing an incomplete sequence in a KStore of claim 1, wherein said KStore includes a plurality of KStore levels having respective current K nodes and said KStore is provided with a state data structure for storing a correspondence between said KStore levels and said current K nodes further comprising first determining said current K node in accordance with said state data structure.

Figure 1


100
TN 465
14
K
11
K Engine
5
API Utility
7
Applications
12
GUI
4
Utility
6
Learn Engine
8
Data Source Applications
13
GUI

1/10

Figure 2A

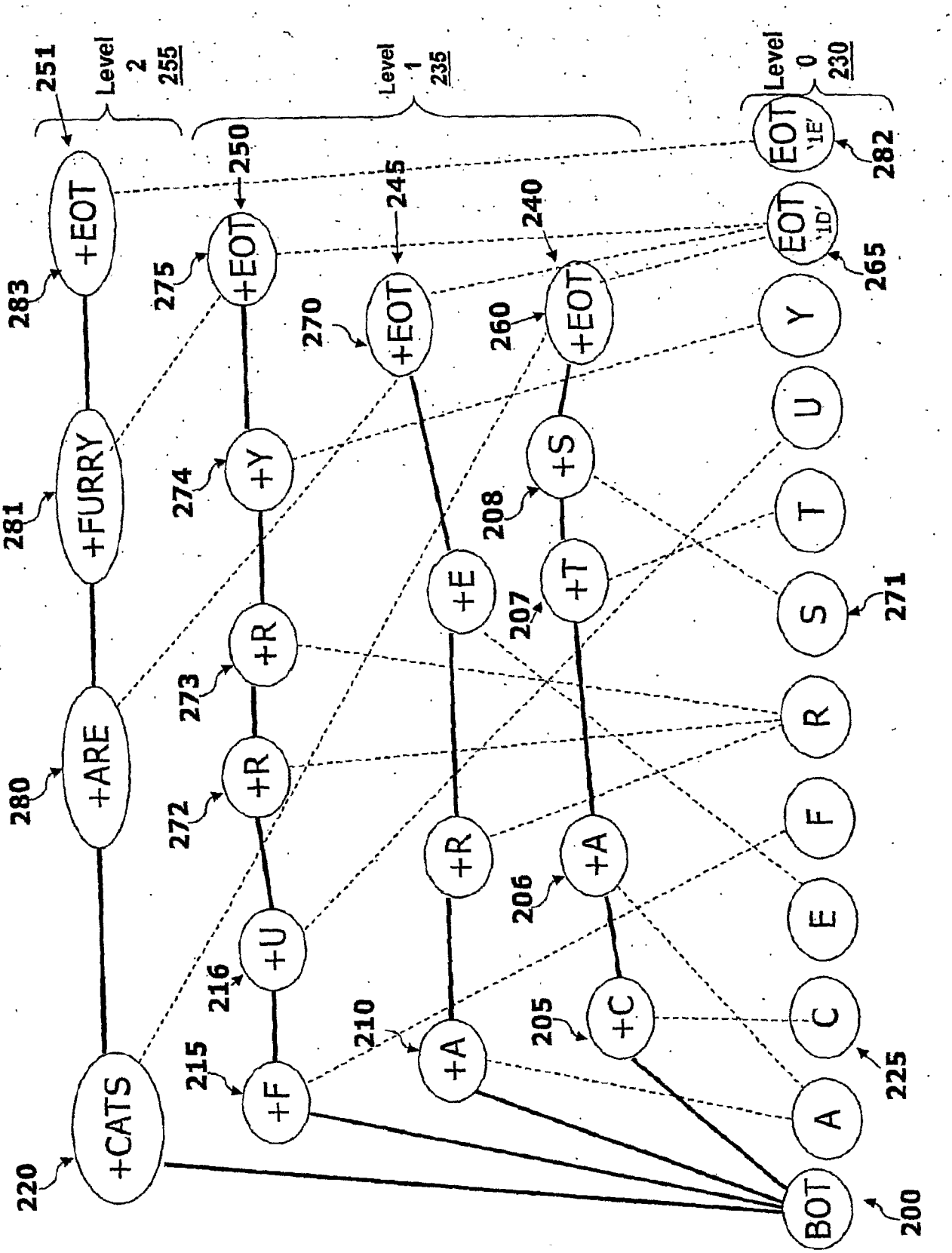
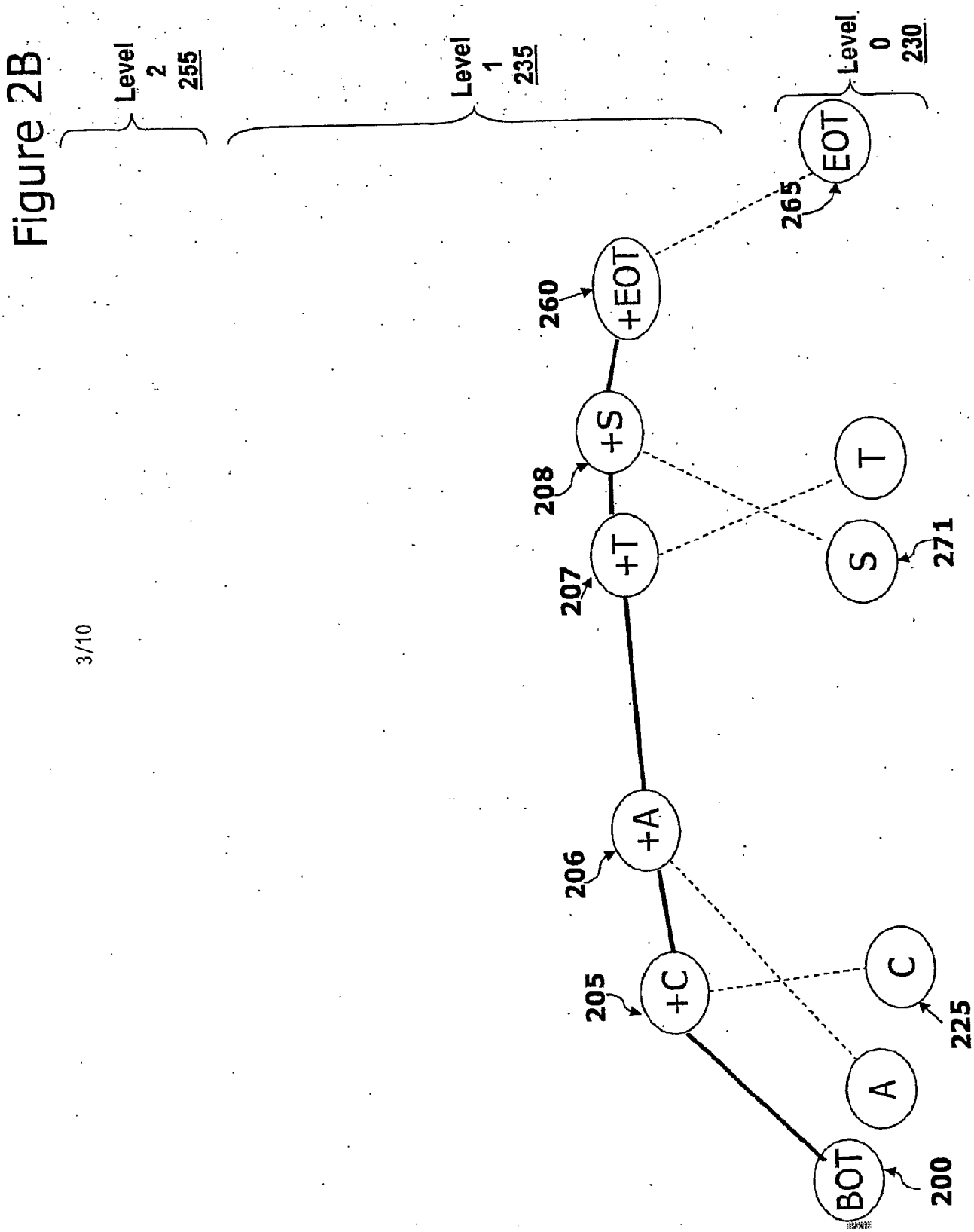
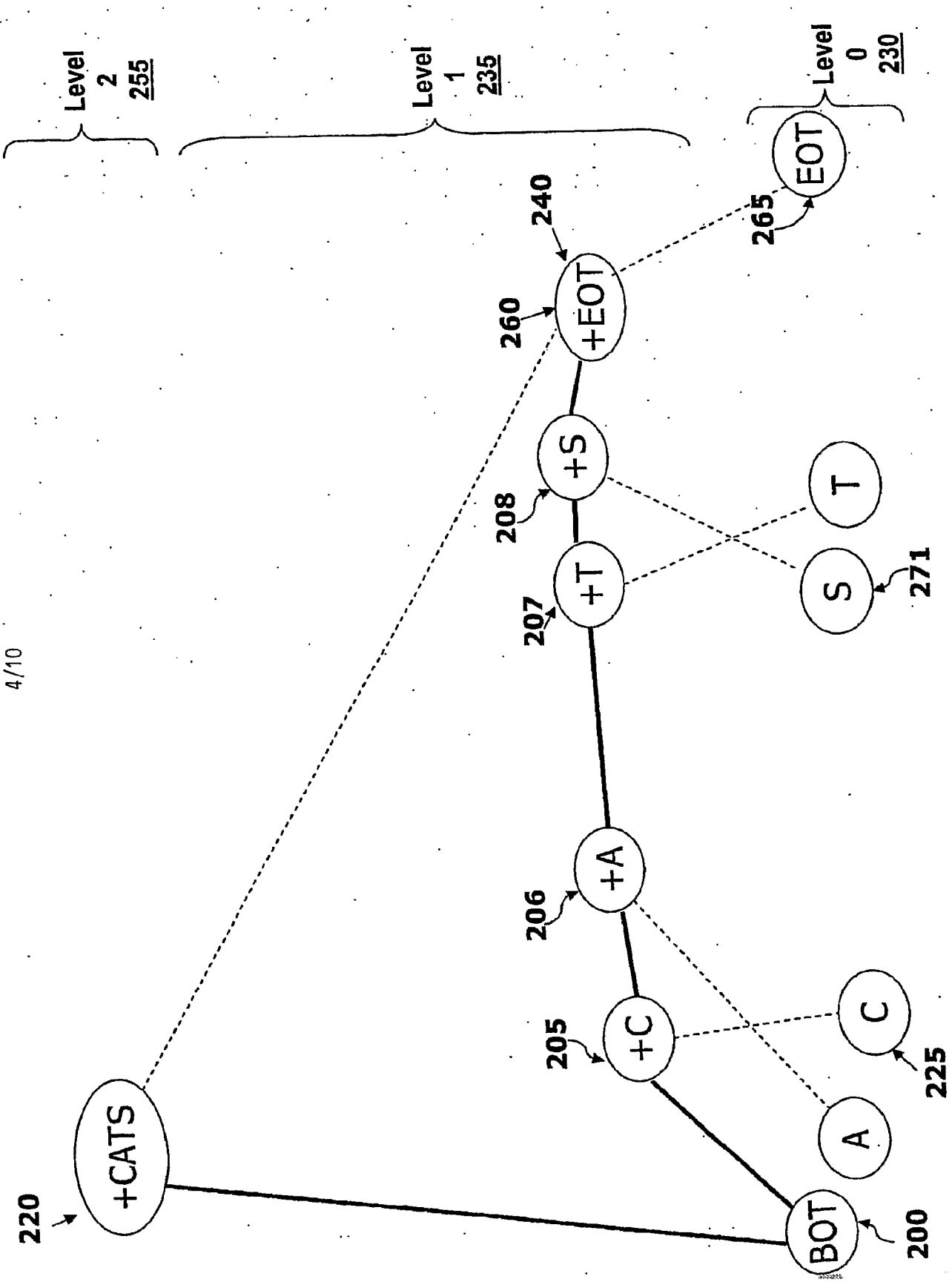

Level 2 255
Level 1 235
Level 0 230
+CATS
+ARE
+FURRY
+EOT
+F
+U
+R
+R
+Y
+EOT
+A
+R
+E
+EOT
+C
+A
+T
+S
+EOT
BOT
A
C
E
F
R
S
T
U
Y
EOT '1D'
EOT '1E'
200
220
280
281
283
215
216
272
273
274
275
210
270
205
206
207
208
240
245
250
251
225
271
265
282

Figure 2B



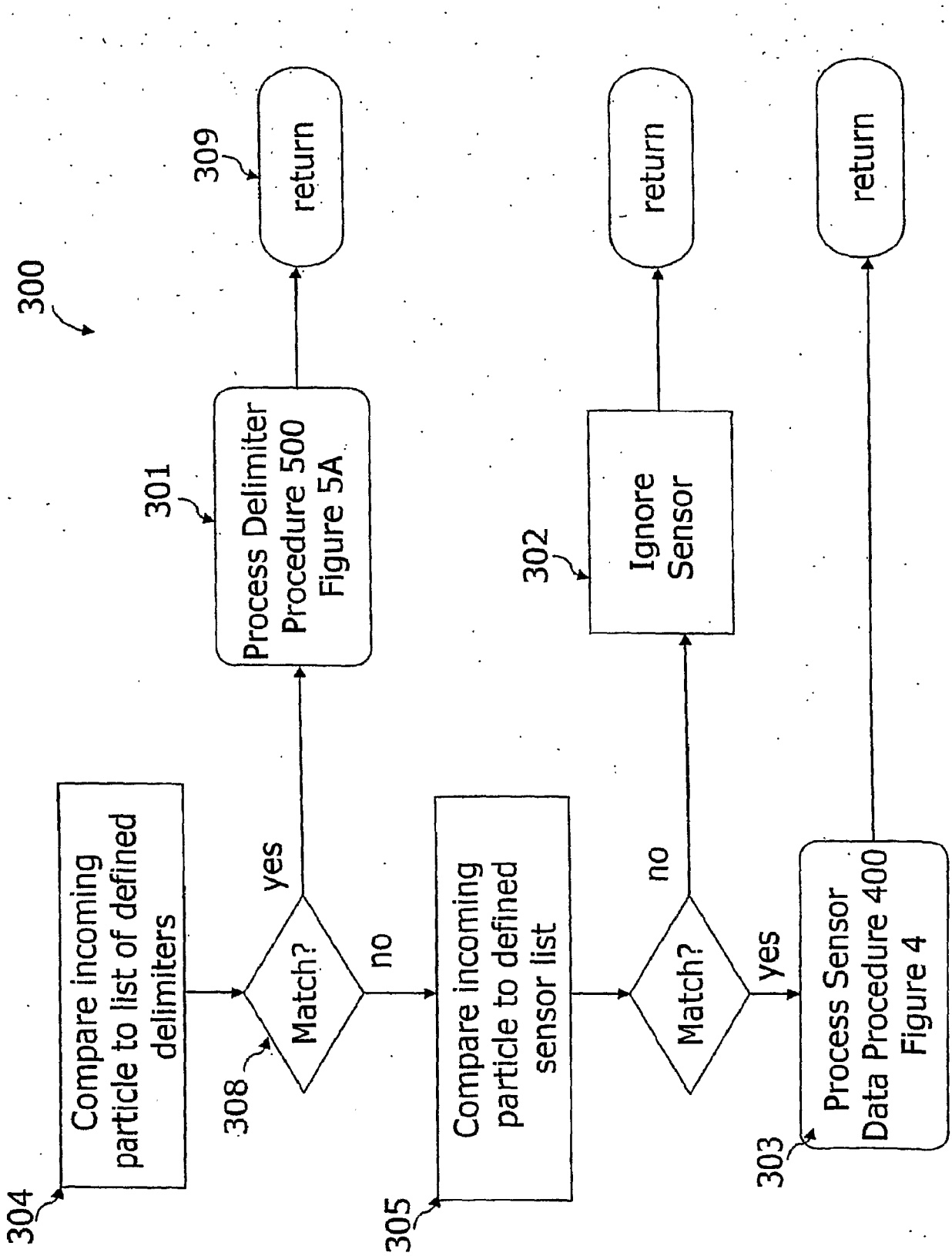
3/10

Figure 2C



4/10

Figure 3


Praxis Procedure
5/10
300
304
Compare incoming particle to list of defined delimiters
308
Match?
yes
301
Process Delimiter Procedure 500 Figure 5A
309
return
no
305
Compare incoming particle to defined sensor list
Match?
no
302
Ignore Sensor
return
yes
303
Process Sensor Data Procedure 400 Figure 4
return

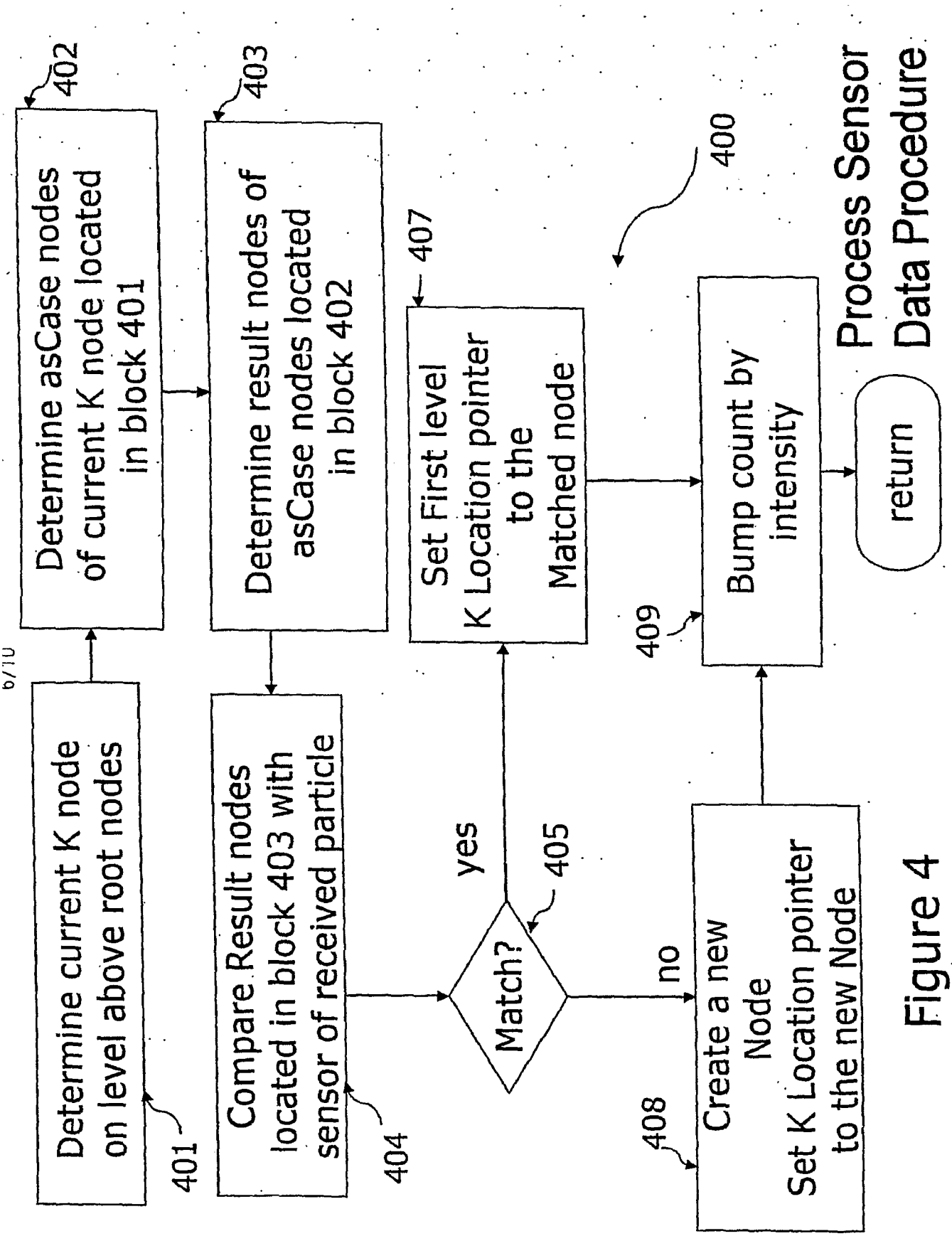
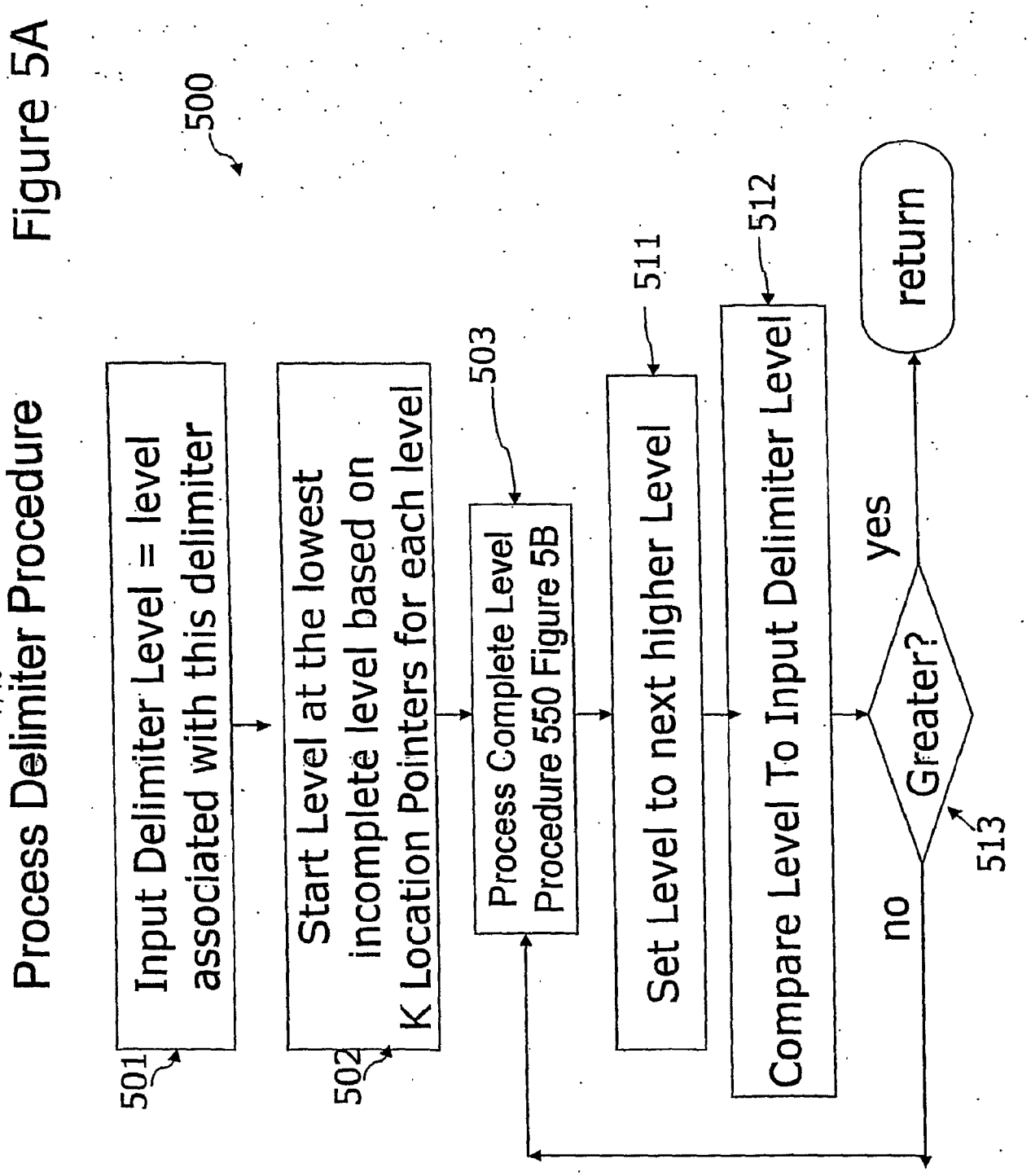
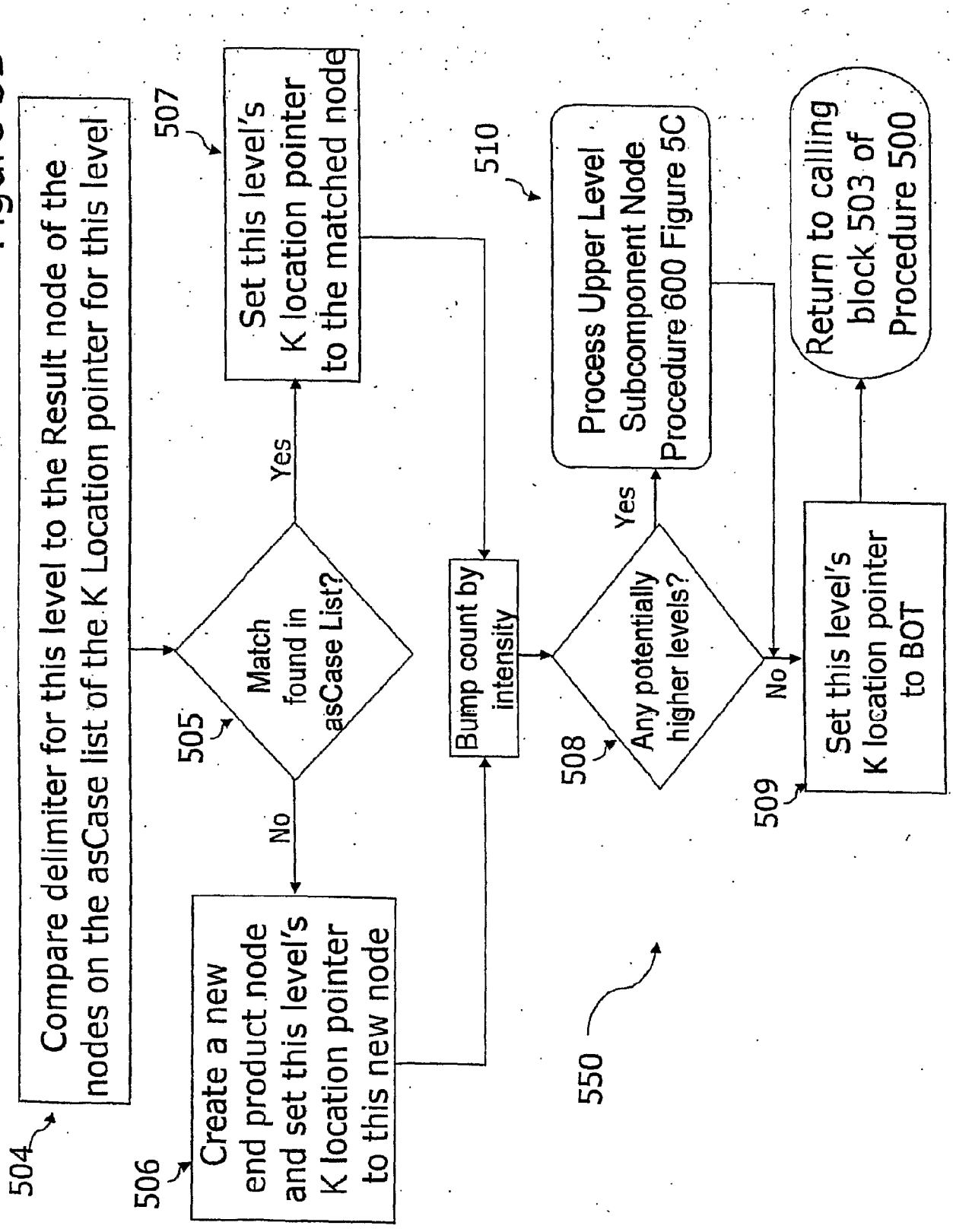

Process Sensor Data Procedure
400
Determine current K node on level above root nodes
401
Determine asCase nodes of current K node located in block 401
402
Determine result nodes of asCase nodes located in block 402
403
Compare Result nodes located in block 403 with sensor of received particle
404
Match?
405
yes
Set First level K Location pointer to the Matched node
407
Bump count by intensity
409
no
Create a new Node
Set K Location pointer to the new Node
408
return

Figure 4

Figure 5A


Process Delimiter Procedure
7/10
500
Input Delimiter Level = level associated with this delimiter
501
Start Level at the lowest incomplete level based on K Location Pointers for each level
502
Process Complete Level Procedure 550 Figure 5B
503
Set Level to next higher Level
511
Compare Level To Input Delimiter Level
512
Greater?
513
yes
no
return


Process Complete Level Procedure
Figure 5B
8/10
Compare delimiter for this level to the Result node of the nodes on the asCase list of the K Location pointer for this level
504
Match found in asCase List?
505
Yes
No
Set this level's K location pointer to the matched node
507
Create a new end product node and set this level's K location pointer to this new node
506
Bump count by intensity
510
Any potentially higher levels?
508
Yes
No
Process Upper Level Subcomponent Node Procedure 600 Figure 5C
Set this level's K location pointer to BOT
509
Return to calling block 503 of Procedure 500
550

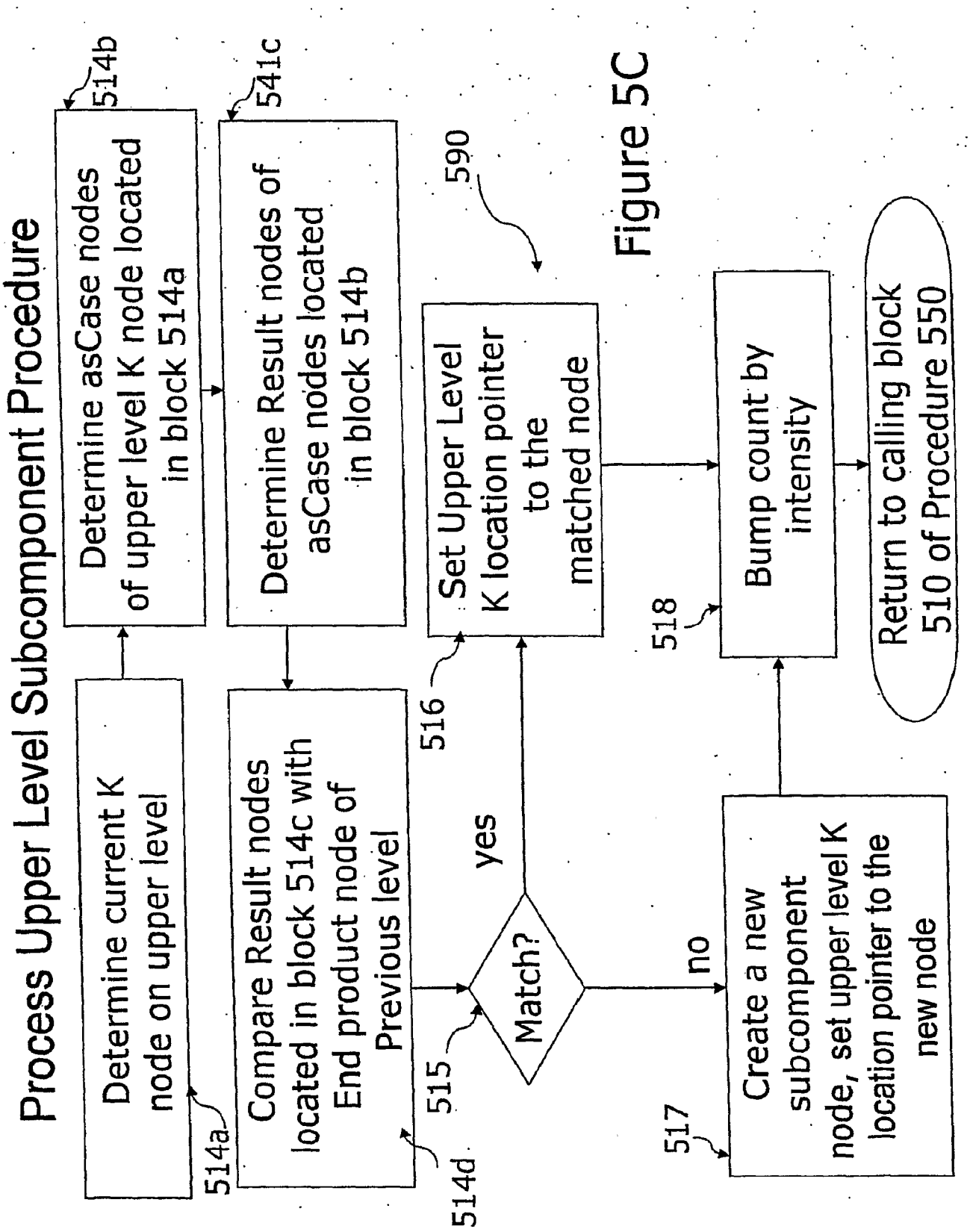

Process Upper Level Subcomponent Procedure
Determine current K node on upper level
514a
Determine asCase nodes of upper level K node located in block 514a
514b
Compare Result nodes located in block 514c with End product node of Previous level
514d
Determine Result nodes of asCase nodes located in block 514b
541c
515
Match?
yes
no
516
Set Upper Level K location pointer to the matched node
590
517
Create a new subcomponent node, set upper level K location pointer to the new node
518
Bump count by intensity
Return to calling block 510 of Procedure 550

Figure 5C

Figure 6A

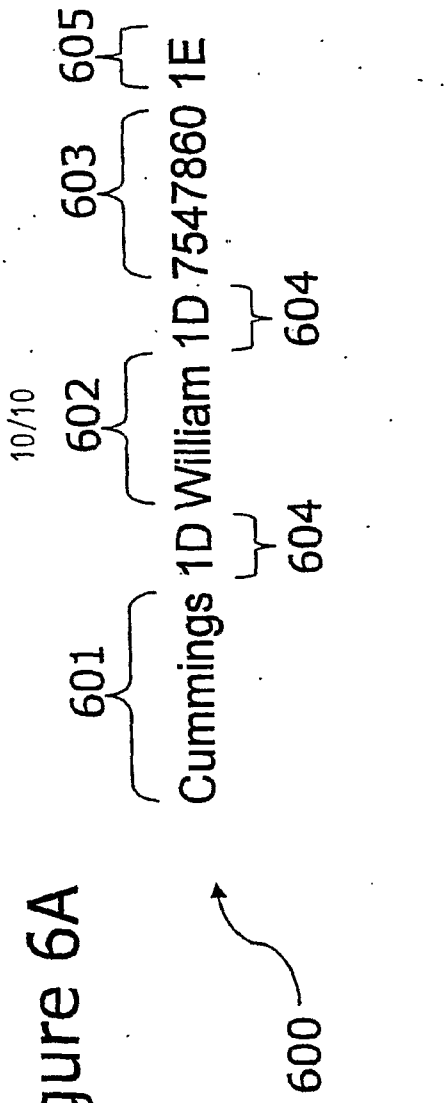


Figure 6B