S&M
Split and Merge Compression Algorithm

By

Abdullah Hashim

The Tree Structure - Coding -

S&M algorithm: The Tree Structure

Coding: is a process of naming the events of an environment Θ by a unique binary codeword.

To code each node v_i of ***EDT***, firstly each set s_i of ***EDT*** is coded by a prefix codeword called ***set codeword*** denoted by $w_{set}(s_i)$. Secondly each node v_i in s_i is coded by a prefix codeword called ***node codeword*** denoted $w_{node}(v_i)$. Each word w_i in the ***ED*** is coded by a ***word codeword*** $w_{word}(w_i)$ which is determined by concatenating the two strings of the set and node codewords.

$w_{word}(w_i) = w_{set}(s_i) + w_{node}(v_i)$, this is a string sum.

The length of the word codeword is therefore equal to the sum of the set and node codeword lengths. Code efficiency is determined by the average codeword length.

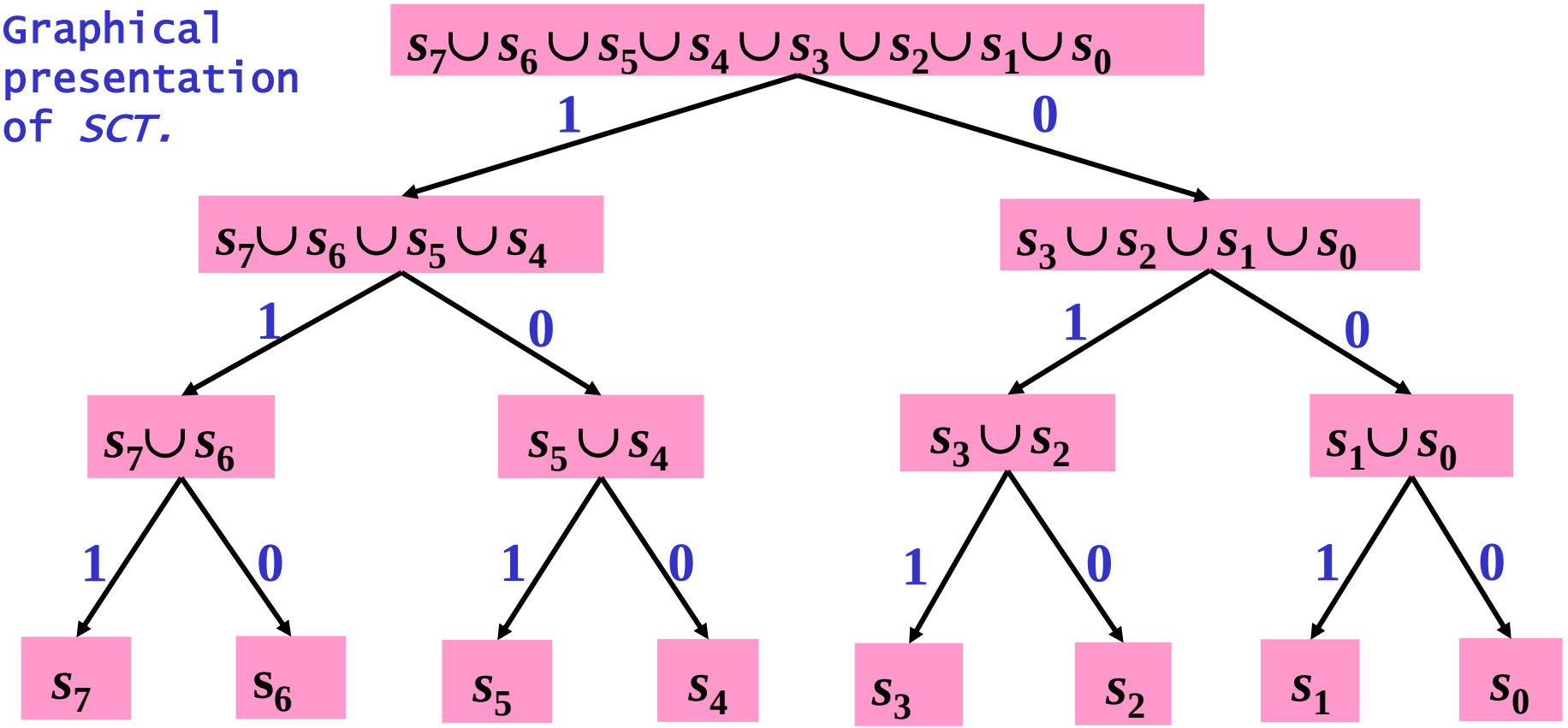
$$L_{average}(ED) = (1 / \Phi) \sum_{over\ \Phi} \mathrm{L}[w_{word}(w_i)]$$ and that of a

compressed file s_{CF}, $L_{average}(s_{CF}) = (1 / |s_{SF}|) \sum_{over\ |s_{CF}|} \mathrm{L}[w_{word}(w_i)]$

S&M algorithm: The Tree Structure

Set-Coding Tree (***SCT***): set-coding tree is a graphical presentation of the ***EDT*** node sets. The links of ***SCT*** presents prefix codeword of sets. Each of the ***SCT*** node contains a set equal to the union of all the sets of its children sets. The root node of ***SCT*** is therefore contains all the sets of ***EDT***. Let ***SCT*** = $<s_7, s_6, s_5, s_4, s_3, s_2, s_1, s_0>$

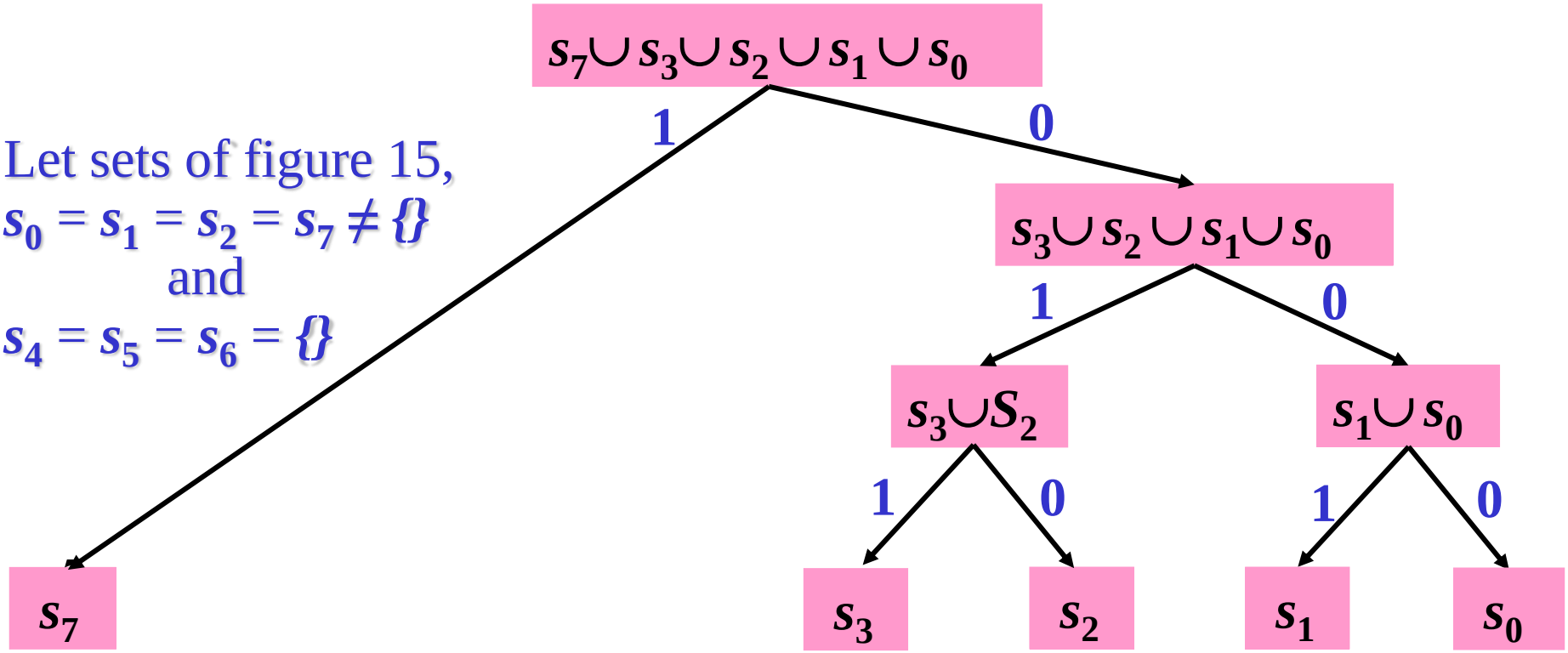
Graphical presentation of *SCT*.



S&M algorithm: The Tree Structure

Empty sets carries no information, nodes and links of empty sets are therefore redundant and should be removed from the ***SCT***.

Let sets of figure 15,
$s_0 = s_1 = s_2 = s_7 \neq \{\}$
and
$s_4 = s_5 = s_6 = \{\}$



s_7 may be coded by the following variable binary code **1**
s_3 may be coded by the following variable binary code **011**
s_2 may be coded by the following variable binary code **010**
s_1 may be coded by the following variable binary code **001**
s_0 may be coded by the following variable binary code **000**

S&M algorithm: The Tree Structure

Set Coding:

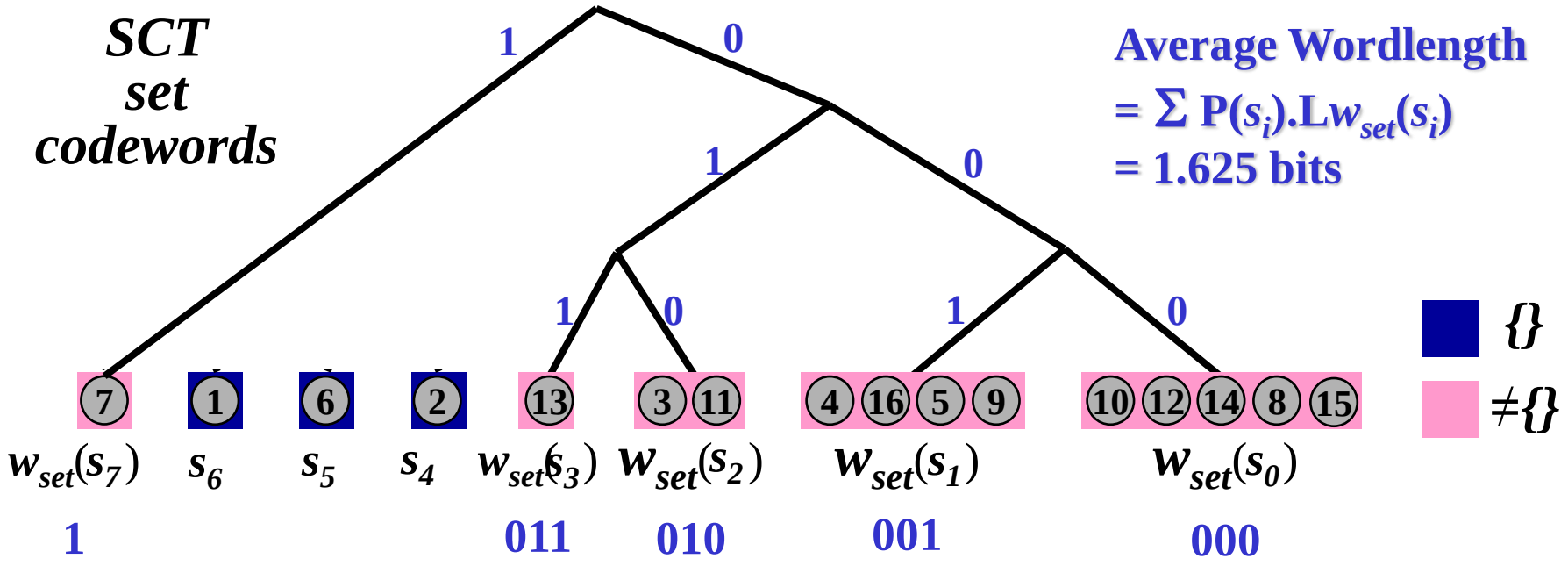
Let an ***OED*** consist of the following words:

$<w_7, w_1, w_6, w_2, w_{13}, w_3, w_{11}, w_4, w_{16}, w_5, w_9, w_{10}, w_{12}, w_{14}, w_8, w_{15}>$ corresponding to the following nodes in ***OEDT*:**

$<v_7, v_1, v_6, v_2, v_{13}, v_3, v_{11}, v_4, v_{16}, v_5, v_9, v_{10}, v_{12}, v_{14}, v_8, v_{15}>$ partitioned into 8 sets: $<s_7, s_6, s_5, s_4, s_3, s_2, s_1, s_0>$; where:

$s_7 = \{v_7\}$; $s_6 = \{v_1\} = \{\}$; $s_5 = \{v_6\} = \{\}$; $s_4 = \{v_2\} = \{\}$; $s_3 = \{v_{13}\}$;

$s_2 = \{v_3, v_{11}\}$; $s_1 = \{v_4, v_{16}, v_5, v_9\}$; and. $s_0 = \{v_{10}, v_{12}, v_{14}, v_8, v_{15}\}$.



S&M algorithm: The Tree Structure

Identifying sets: *Set links* (***SL***) are used to identify a set s_i and determine its codeword $\{w_{set}(s_i)\}$. ***SL*** connects the first node say (***m***) to the last node say (***n***) of a given set or the union of (2^r) adjacent sets in an ***OEDT***, (where ***r*** is a positive integer). If node ***n*** is on the right of node ***m*** it is called a right set-link (***RSL***). If node ***n*** is on the left of node ***m*** the link is said to be a left set-link (***LSL***). Nodes surrounded by a link are the nodes of a given set or union of sets. The set-link is said to be of height (***h***) and is denoted by ***SL***(***h***) if it surrounds the nodes of the union of two set-links of height (***h*-1)** denoted by ***SL***(***h* -1**). The set-link of height zero ***SL***(**0**) surrounds all the nodes of a single set s_i. The number of set-links of height ***h*** is twice the number of set-links of height ***h*+1**. The ***i***-*th* right set-link denoted by ***RSL***$(h)_i$ and the corresponding ***i***-*th* left set-link denoted ***LSL***$(h)_i$ surrounds the same nodes. Each of the set-links is coded by a single binary digit. The zero height set-link has the least significant digit and the largest height set-link has the most significant digit in a set's codeword $w_{set}(s_i)$. Set-links are of two types. The first (***type* 1**) contains at least one data node and it is coded by a single binary digit. The second (***type* 0**) is redundant and contains only no-data, and surround a ***null*** nodes.

S&M algorithm: The Tree Structure

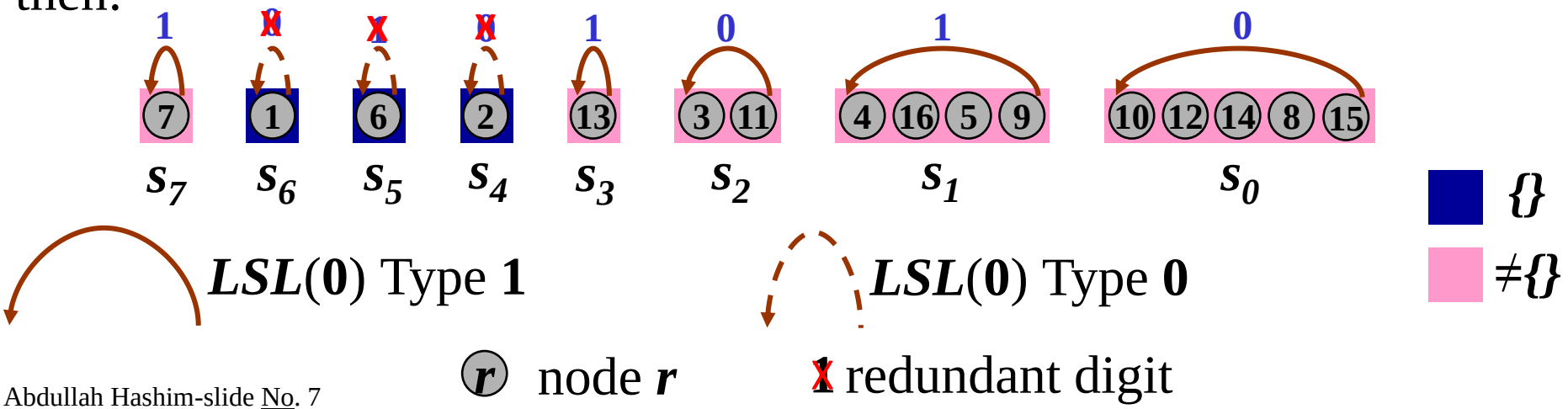
Left set-links of height zero LSL(0)$_i$: is a set link connecting the rightmost to the leftmost node of a single set s_i. ***RSL(0)$_i$*** connect the leftmost to the rightmost node of s_i.

Let an ***OED*** consist of the following words:

$<w_7, w_1, w_6, w_2, w_{13}, w_3, w_{11}, w_4, w_{16}, w_5, w_9, w_{10}, w_{12}, w_{14}, w_8, w_{15}>$ corresponding to the following nodes in ***OEDT*:**

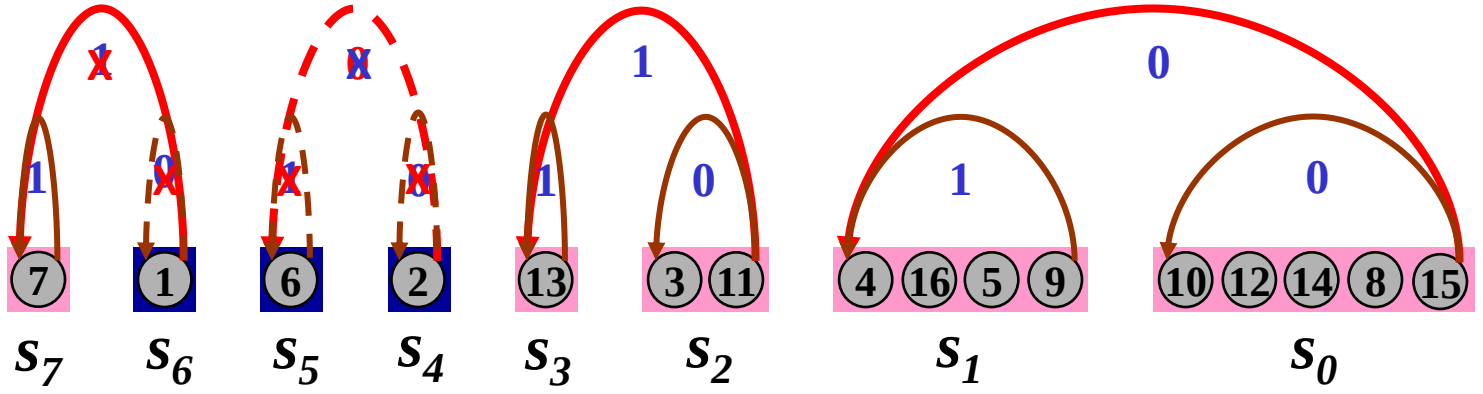
$<v_7, v_1, v_6, v_2, v_{13}, v_3, v_{11}, v_4, v_{16}, v_5, v_9, v_{10}, v_{12}, v_{14}, v_8, v_{15}>$ partitioned into 8 sets: $<s_7, s_6, s_5, s_4, s_3, s_2, s_1, s_0>$; where:

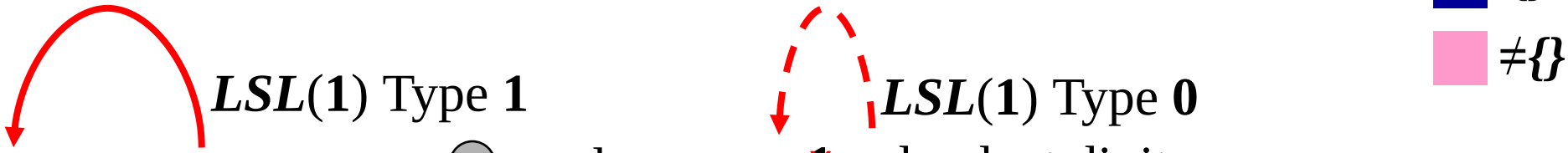
$s_7 = \{v_7\}$; $s_6 = \{v_1\} = \{\}$; $s_5 = \{v_6\} = \{\}$; $s_4 = \{v_2\} = \{\}$; $s_3 = \{v_{13}\}$; $s_2 = \{v_3, v_{11}\}$; $s_1 = \{v_4, v_{16}, v_5, v_9\}$; and. $s_0 = \{v_{10}, v_{12}, v_{14}, v_8, v_{15}\}$; then:



S&M algorithm: The Tree Structure

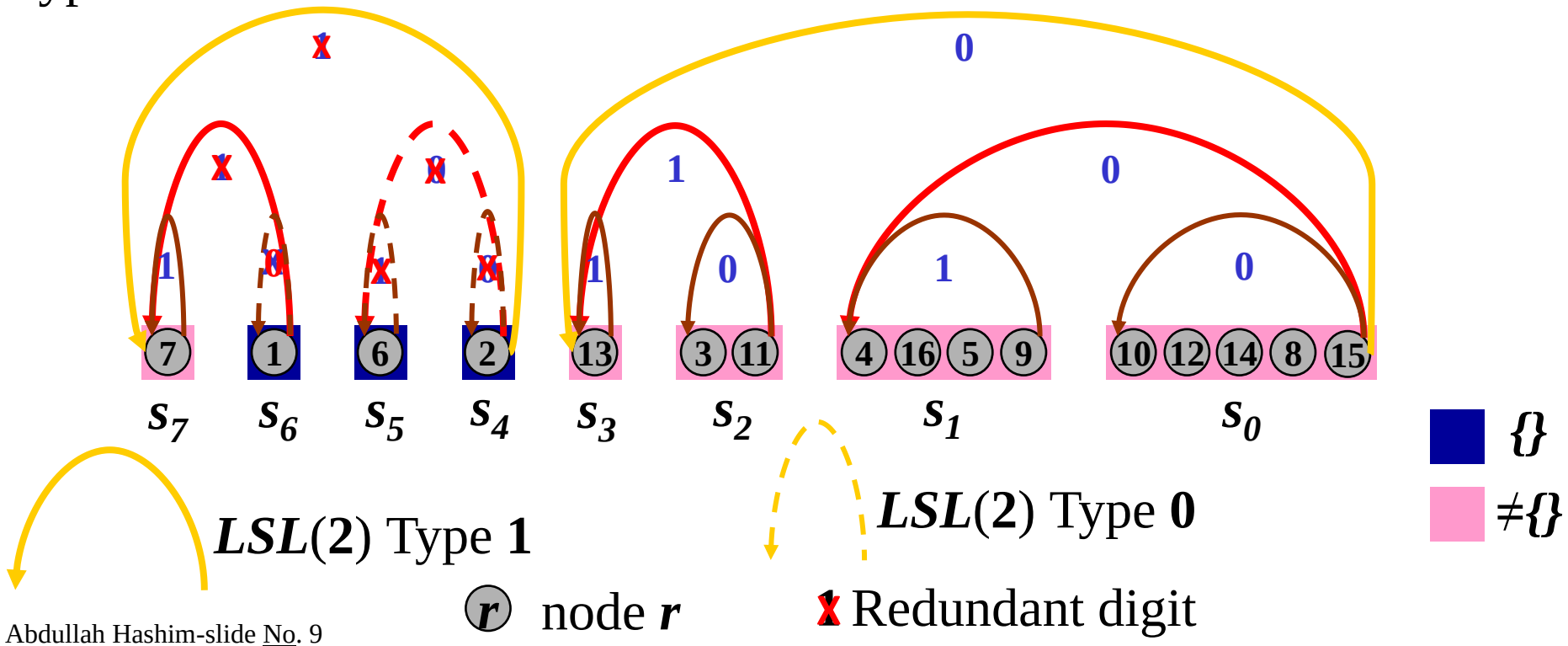
Left set-links of height one {$LSL(1)_i$}: is a set-link connecting the rightmost node of the (***2i***)-***th*** left-set-link {$LSL(0)_{2i}$} to the leftmost node of the (***2i+1***)-*th* left-set-link {$LSL(0)_{2i+1}$}. If $LSL(0)_{2i+1}$ is a null-link then $LSL(1)_i$ surrounds only nodes of $LSL(0)_{2i}$ and (in this case) $LSL(0)_{2i}$ becomes redundant. $RSL(1)_i$ surrounds the same nodes of $LSL(1)_i$. If one of the two $SL(0)$ set-links is of type **0**, then the $SL(0)$ type **1** becomes redundant. $SL(1)$ type **1** surrounds at least one $SL(0)$ of type **1**. $SL(1)$ type **0** surrounds only type **0** {$SL(0)$} set-links.





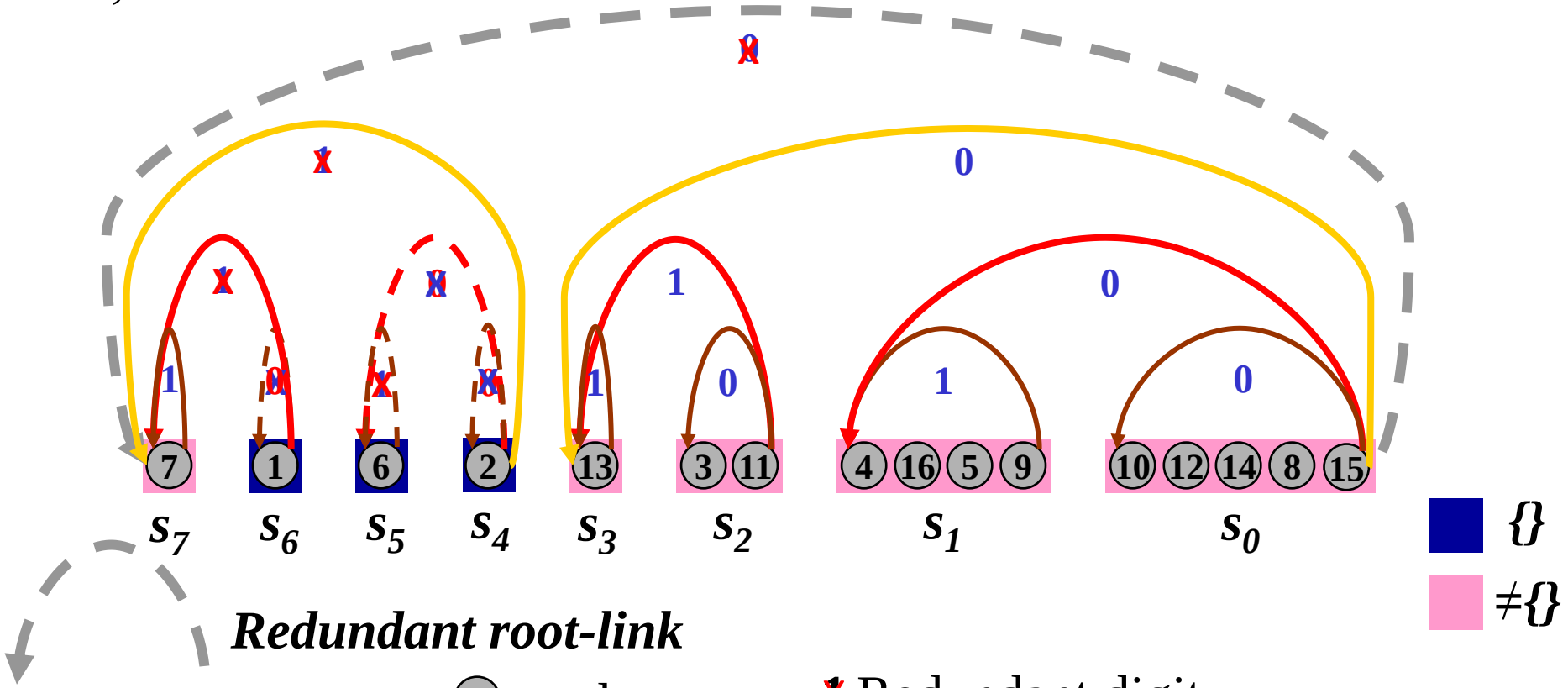
S&M algorithm: The Tree Structure

Left set-links of height two $\{LSL(2)_i\}$**:** is a set-link connecting the rightmost node of the (***2i***)-***th*** left-set-link $\{LSL(1)_{2i}\}$ to the leftmost node of the (***2i+1***)-*th* left-set-link $\{LSL(1)_{2i+1}\}$. If $LSL(1)_{2i+1}$ is a null-link then $LSL(2)_i$ surrounds only nodes of $LSL(1)_{2i}$ and (in this case) $LSL(1)_{2i}$ becomes redundant. $RSL(2)_i$ surrounds the same nodes of $LSL(2)_i$. If one of the two $SL(1)$ set-links is of type **0**, then the $SL(1)$ type **1** becomes redundant.



S&M algorithm: The Tree Structure

Left set-links of height three $\{LSL(3)_i\}$**:** is a set-link connecting the rightmost node of the (***2i***)-***th*** left-set-link $\{LSL(2)_{2i}\}$ to the leftmost node of the (***2i+1***)-***th*** left-set-link $\{LSL(2)_{2i+1}\}$. If $LSL(1)_{2i+1}$ is a null-link then $LSL(3)_i$ surrounds only the nodes of $LSL(2)_{2i}$ and (in this case) $LSL(2)_{2i}$ becomes redundant. A link surrounding all nodes of all sets, it is redundant an called root set-link.



S&M algorithm: The Tree Structure

Notes on Set Coding Tree *SCT* :

1 $SL(h)_i$ is the ***i***-th set-link of height ***h***, where ***i*** =1, 2, …, ***r***;
$r = \lfloor N / 2^h \rfloor$, and ***N*** is the number of sets in ***SCT***. $SL(h)_I$ surrounds the nodes of set-links $SL(h\text{-}1)_{2i}$ and $SL(h\text{-}1)_{2i+1}$. If $SL(h\text{-}1)_{2i+1}$ is a null-link then $SL(h)_i$ surrounds only the nodes of $SL(h\text{-}1)_{2i}$ and (in this case) $SL(h\text{-}1)_{2i}$ becomes redundant.

2 Links pointing from left to right are called right links, and links pointing from right to left are called left links.

3 $LSL(h)_i$ and the $RSL(h)_i$ surrounds the same nodes.

4 set-links have a single binary digit code, the ***i***-th set-link is coded by binary zero if ***i*** is an even integer and binary one if ***i*** is an odd integer. The zero height set-link has the least significant digit and the largest height set-link has the most significant digit in the set's codeword. A set-link holding no data is redundant and has no code.

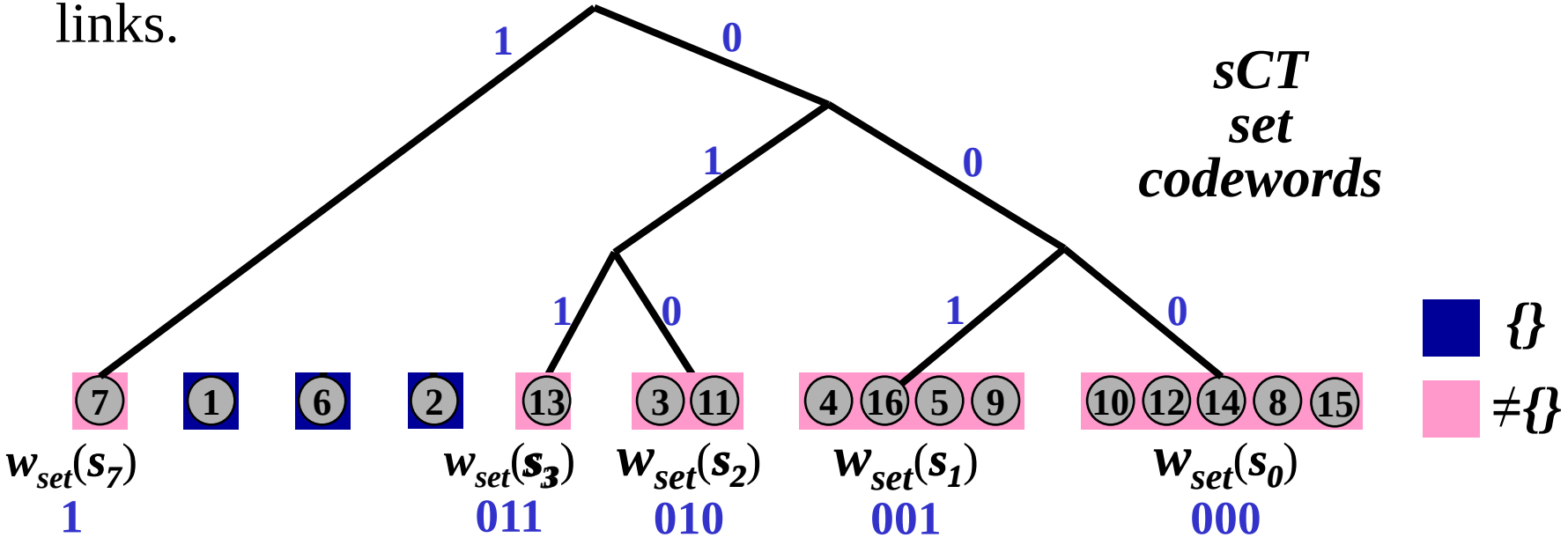
5 A link surrounding all nodes of all sets is called root set-link and is redundant. [$SLd(SCT)$], where $d(SCT)$ is the depth of ***SCT*** denoted by the letter (***t***);
$t = \lfloor \log_2 N \rfloor$ is known as the root-link.

S&M algorithm: The Tree Structure

6. ***SL*(*0*)** surrounds the nodes of a single set.

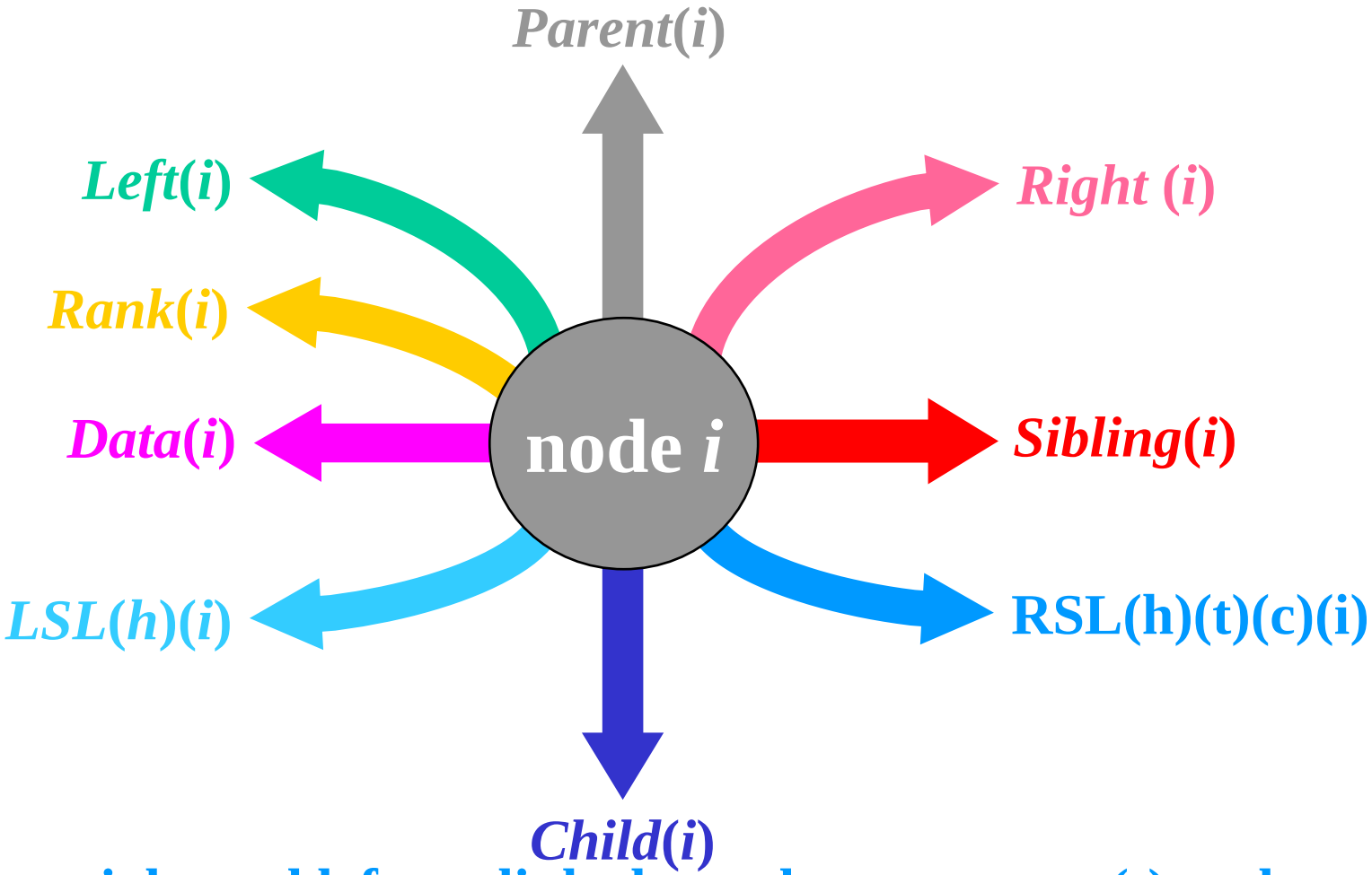
7. set-links are of two types, type **1** surrounds at least one data node, while type **0** surrounds only no-data nodes. A type **1** set-link is coded by a single binary digit, while type **0** is a redundant link.

8. If ***SL*(*h*+1)** surrounds all nodes of two adjacent ***SL*(*h*)** set-links and if one of the two ***SL*(*h*)** set-links is of type **0**, then the ***SL*(*h*)** type **1** becomes redundant. ***SL*(*h*+1)** type **1** surrounds at least one ***SL*(*h*)** of type **1**. ***SL*(*h*+1)** type **0** surrounds only type **0** ***SL*(*h*)** set-links.



S&M algorithm: The Tree Structure

ODT* links**: The coding digit (c***) may take three values, 0, 1 for binary zero and one respectively and 2 for redundant digit.

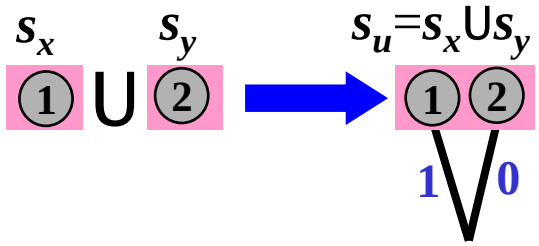


Since right and left set-links have the same type (t) and coding digit (c), left set-link type and code digit is ignored.

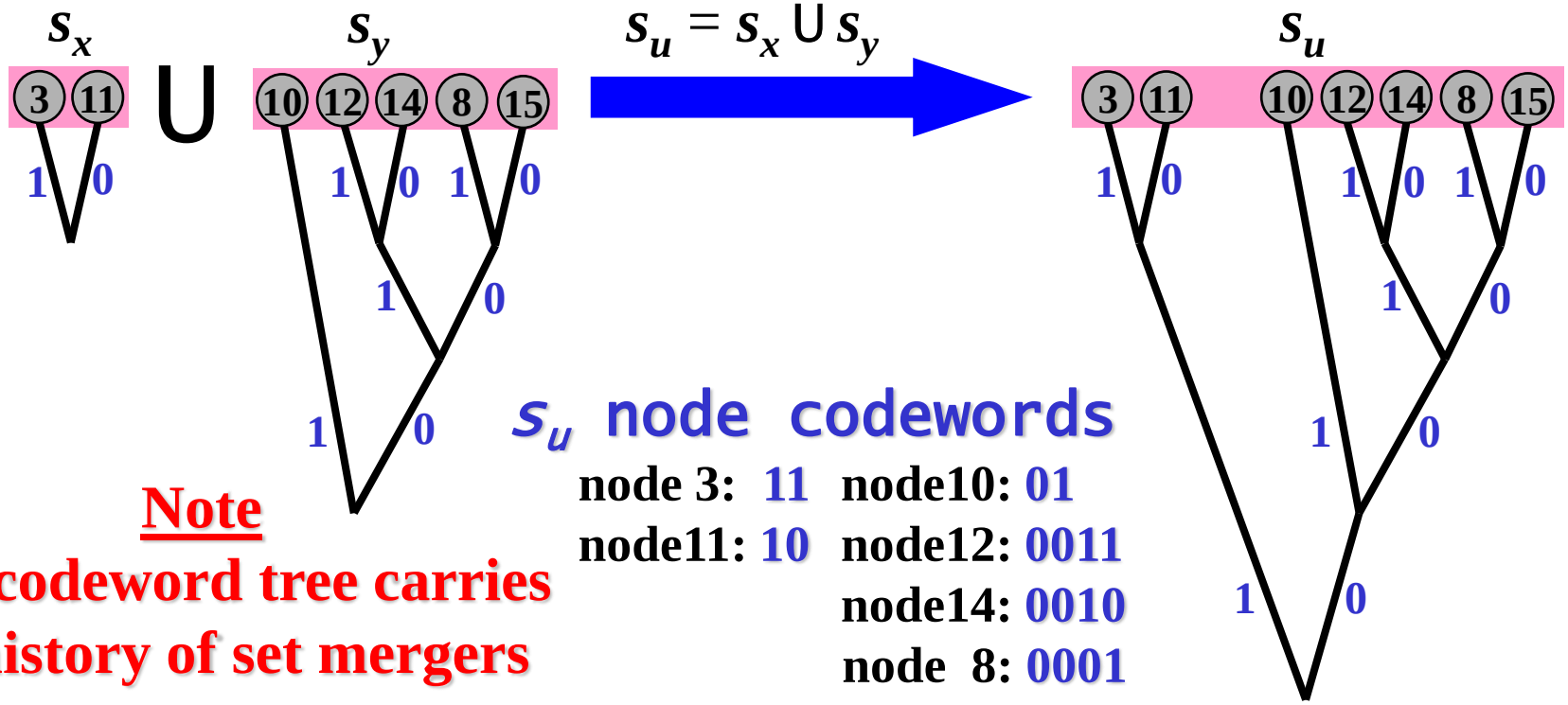
S&M algorithm: The Tree Structure

Node Coding:

When two singleton s_x and s_y sets merge into one set s_u, the two nodes in s_u are coded by a single binary digit (1) and (0) respectively.



similarly, the merger of two multi nodes sets, s_x and s_y into set (s_u), $s_u = s_x \cup s_y$, if $|s_x| = 2$, $|s_y| = 5$, then $|s_u| = 7$, nodes of s_x coded by binary digit (0) and that of s_y by (1), as shown below.



Note

node codeword tree carries the history of set mergers

s_u node codewords

node 3: 11 **node10: 01**
node11: 10 **node12: 0011**
node14: 0010
node 8: 0001
node05: 0000

S&M algorithm: The Tree Structure

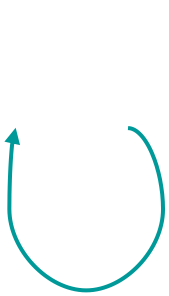
Node coding: Initially the dictionary contains α single character words corresponding to the alphabet in A and χ single character words corresponding to control elements in C. All sets in the initial tree have zero length node codewords. Multiple node sets are formed by the successive merging process. Assuming the two sets to be merged have equal probability and a single binary digit will be appended to the merged set node codewords. The node codewords of the first set of the merger are appended by one, while those of the second set of the merger are appended by zero. This process ensures that the merged set of nodes are optimally coded. To ensure robustness, in sets with node codewords of length equal to a given Ψ have their NCT code bounded to keep the maximum height within the bound.

Node codewords (w_{node}) are appended to their corresponding set codewords (w_{set}) to form the compressed word codewords (w_{word}).

S&M algorithm: The Tree Structure

Node Coding: ***node-links (NL)*** are used to determine node codeword $w_{node}(v_i)$ for a node v_i in set s_i. ***NL*** is a link between nodes within a set. It has the same structure as the ***SL***; it consists of right and left node-links, {***RNL*(*h*)**} and {***LNL*(*h*)**} respectively, of height (***h***) (where $h = 0, 1, \ldots, h_{max}$, and $\lfloor \log_2 (|s_i|_{max}) \rfloor \leq h_{max} \leq (\Phi - N)$). $RNL(h)_i$ and $LNL(h)_i$ surrounds the same nodes. Node-links have no type unlike the case of ***SL***. A link containing all nodes in a set s_i and its corresponding node-link $NL(d_{NCT})$, (where d_{NCT} is the binary ***NCT*** depth) are redundant. $NL(d_{NCT})$ is known as the root node-link. All single node sets have only root node-links.

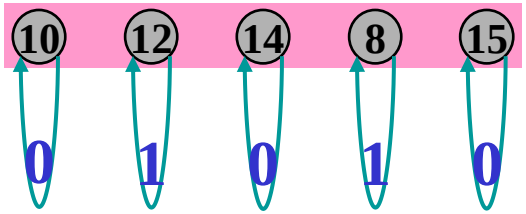
Node-links of height zero **{*NL*(0)}:** is a node-link surrounding a single node. The ***LNL*(0)** and ***RNL*(0)** surrounds the same single node. Consider set s_0 of the previous examples: $s_0 = \{v_{10}, v_{12}, v_{14}, v_8, v_{15}\}$.



***LNL*(0)**

node ***r***

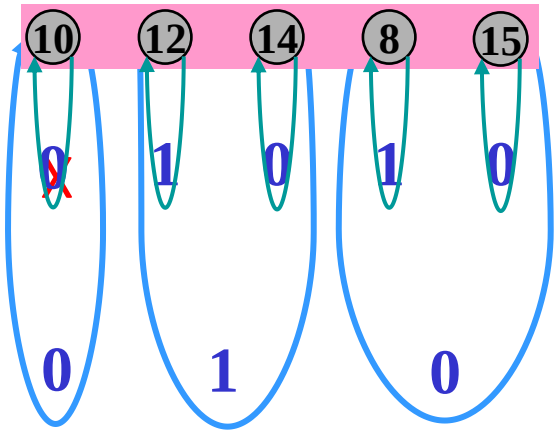
***NCT* of s_0; *h* = 0**



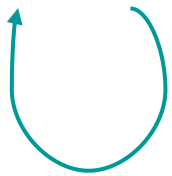
S&M algorithm: The Tree Structure

Left node-link of height One $\{LNL(1)_i\}$**:** is a node-link connecting the rightmost node of the (***2i***)-***th*** left-node-link $\{LNL(0)_{2i}\}$ to the leftmost node in the (***2i+1***)-***th*** left-node-link $\{LNL(0)_{2i+1}\}$. If $LNL(0)_{2i+1}$ is a null-link then $LNL(1)_i$ surrounds the only single node of $LNL(0)_{2i}$ and (in this case) $LNL(0)_{2i}$ becomes redundant. $RNL(1)_i$ and $LNL(1)_i$ contain the same nodes.

NCT **of** s_0**;** h **= 0 and 1**



1 Redundant digit



$LNL(1)$

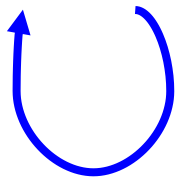
(r) node ***r***

S&M algorithm: The Tree Structure

Left node-link of height two $\{LNL(2)_i\}$**:** is a node-link connecting the rightmost node of the (***2i***)-**th** left-node-link $\{LNL(1)_{2i}\}$ to the leftmost node in the (***2i+1***)-**th** left-node-link $\{LNL(1)_{2i+1}\}$. If $LNL(1)_{2i+1}$ is a null-link then $LNL(2)_i$ surrounds all nodes of $LNL(1)_{2i}$ and (in this case) $LNL(1)_{2i}$ becomes redundant. $RNL(2)_i$ and $LNL(2)_i$ surrounds the same nodes.

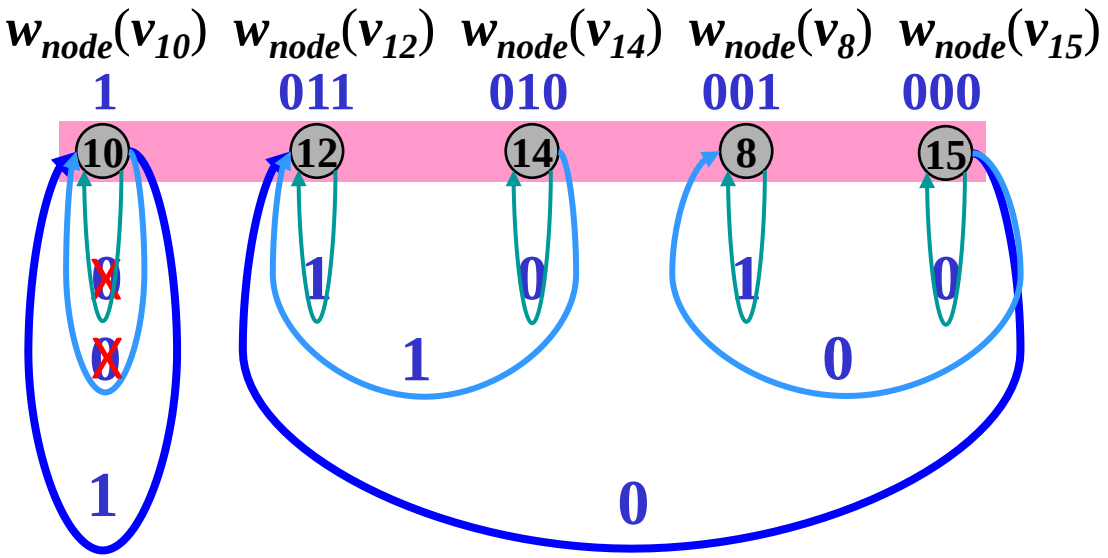
Node	Codeword
10	1
12	011
14	010
8	001
15	000

~~1~~ Redundant digit



$LNL(2)$

Set S_1: node codewords $w_{node}(v_i)$

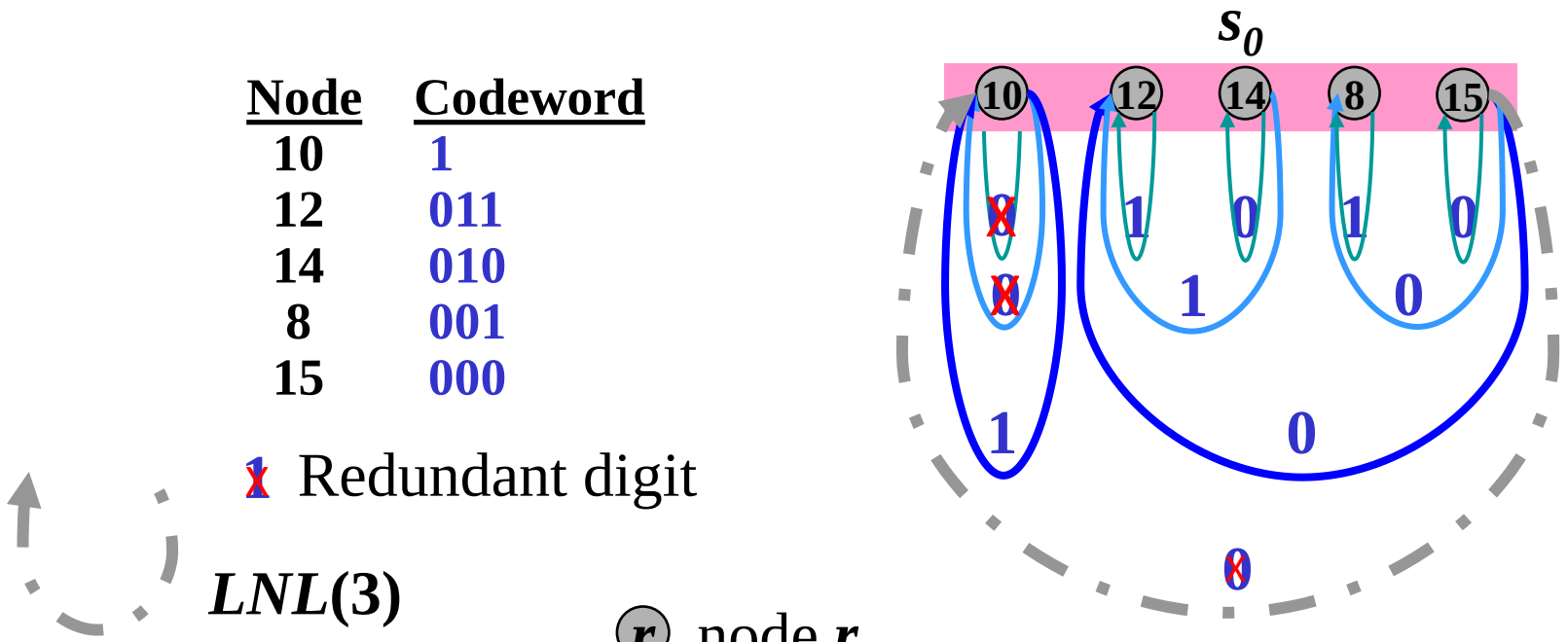


(r) node ***r***

S&M algorithm: The Tree Structure

Left node-link of height three $\{LNL(3)_i\}$**:** is a node-link connecting the rightmost node of the (***2i***)**-th** left-node-link $\{LNL(2)_{2i}\}$ to the leftmost node of the (***2i+1***)-th left-node-link $\{LNL(2)_{2i+1}\}$. If $LNL(2)_{2i+1}$ is a null-link then $LNL(3)_i$ surrounds all nodes of $LNL(2)_{2i}$ and (in this case) $LNL(2)_{2i}$ becomes redundant. Right node-links $RNL(3)_i$ surrounds the same nodes of left node-links $LNL(3)_i$. A link containing all nodes of a set is redundant, known as the root node-link.

Node	Codeword
10	1
12	011
14	010
8	001
15	000



S&M algorithm: The Tree Structure

Notes on Node Coding Tree *NCT*:

1. The i-th node-link ($NL(h)_i$) surrounds the nodes in a set (s_i) is said to be of height h, where $i = 0, 1, \ldots, m-1$; $m = \lfloor |s_i|/2^h \rfloor$. $NL(h)_i$ surrounds the nodes of node-links $NL(h-1)_{2i}$ and $NL(h-1)_{2i+1}$. If $NL(h-1)_{2i+1}$ is a null-link then $NL(h)_i$ surrounds only the nodes of $NL(h-1)_{2i}$ and (in this case) $NL(h-1)_{2i}$ becomes redundant.

2 Links pointing from left to right are called right links, and links pointing from right to left are called left links.

3 $LNL(h)_i$ and the $RNL(h)_i$ surrounds the same nodes.

4 Node-links have a single binary digit code, the i-th node-link is coded by binary zero if i is an even integer and binary one if i is an odd integer. The zero height node-link has the least significant digit and the largest height node-link has the most significant digit in the node codeword $w_{node}(s_i)$.

S&M algorithm: The Tree Structure

Notes on Node Coding Tree *NCT* (Continued):

5 A link surrounding all nodes of a set s_i is denoted by ($NL(d_{NCT})$, (where d_{NCT} is the binary ***NCT*** depth) are redundant. $NL(d_{NCT})$ is known as the root node-link.

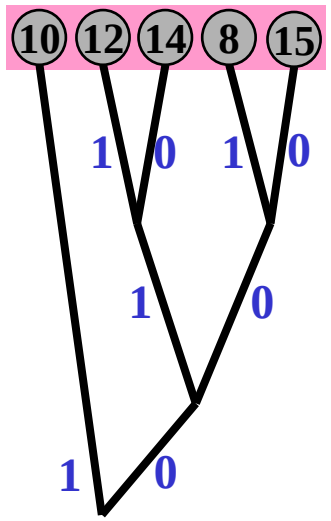
6. ***NL*(0)** surrounds a single node.

7. A node in a ***singleton*** set has a redundant root node-link ***NL*(0)**.

8. ***NCT*** is constructed from right to left (***i*** = 0, 1, 2, …).

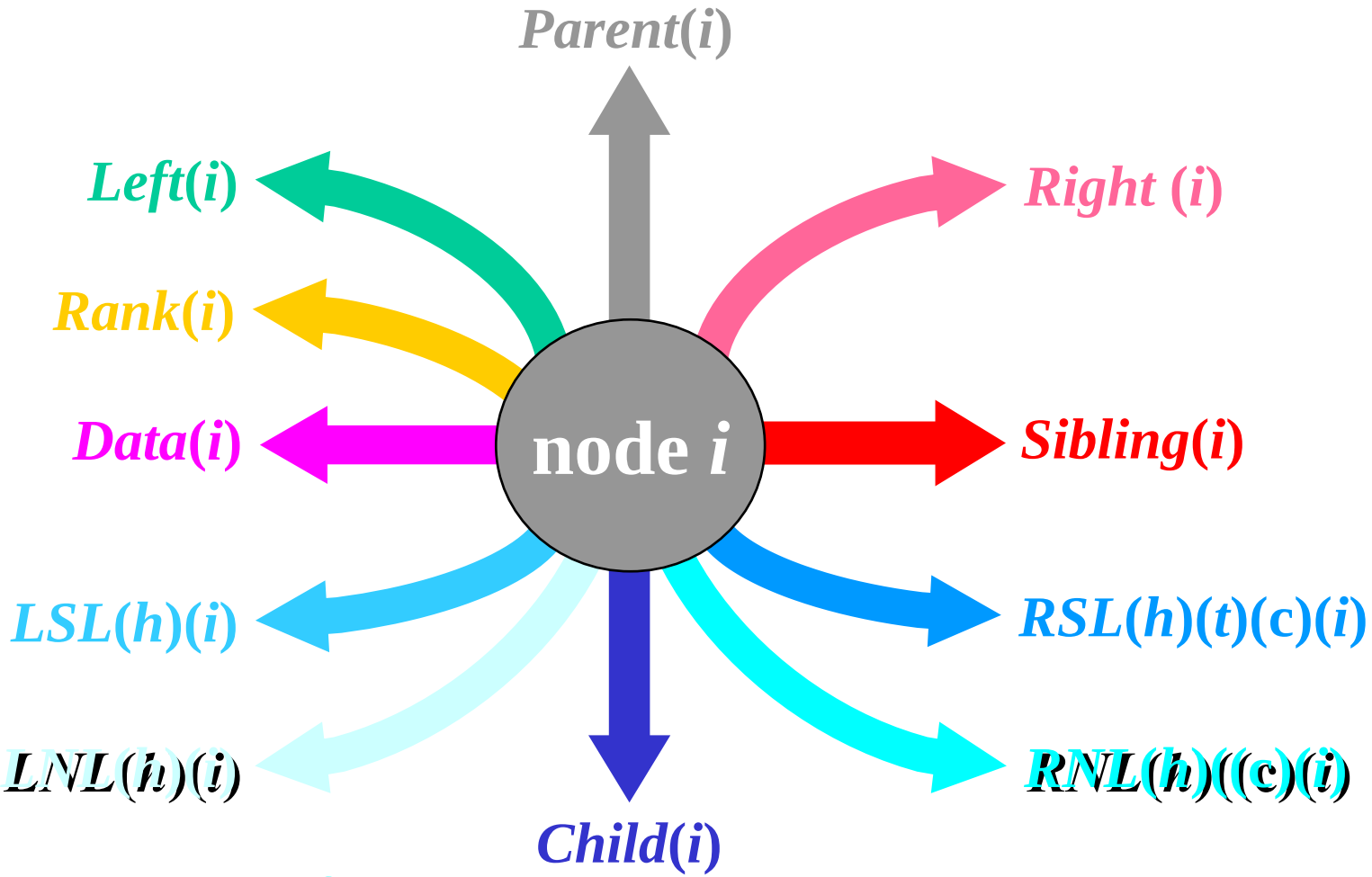
NCT node codewords

Node	Codeword
10	1
12	011
14	010
8	001
15	000



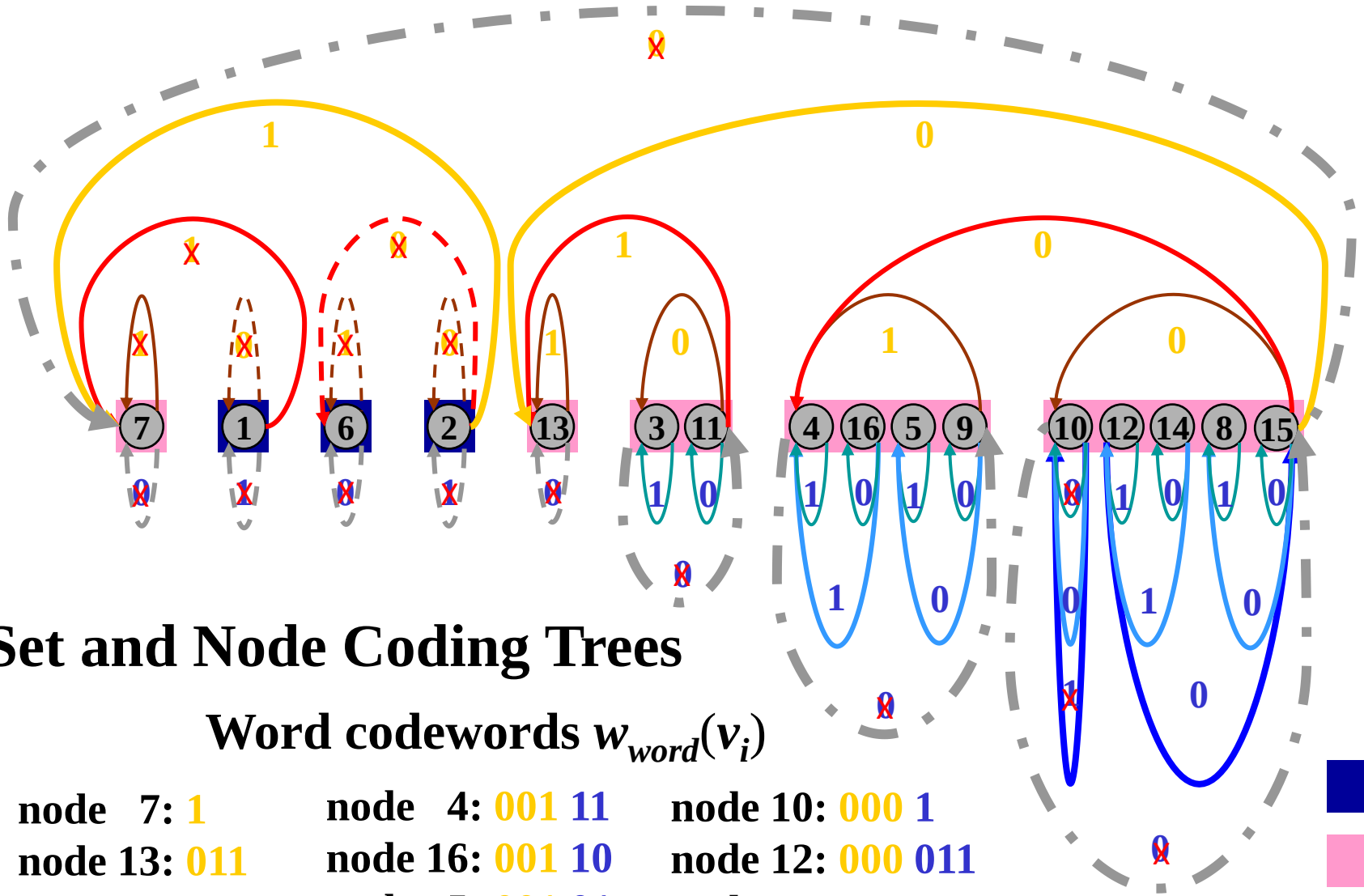
S&M algorithm: The Tree Structure

ODT* links**: The coding digit (c***) may take three values, 1, 1 for binary zero and one respectively and 2 for redundant digit.



Since right and left node-links have the same coding digit (c), left node-link code digit is ignored.

S&M algorithm: The Tree Structure



Set and Node Coding Trees

Word codewords $w_{word}(v_i)$

node 7: 1
node 13: 011
node 3: 010 1
node 11: 011 0
node 4: 001 11
node 16: 001 10
node 5: 001 01
node 9: 001 00
node 10: 000 1
node 12: 000 011
node 14: 000 010
node 8: 000 001
node 15: 000 000

{}
≠{}

S&M algorithm: The Tree Structure

The word codeword

$w_{word} = w_{set}(s_i) + w_{node}(v_i)$

String addition

Average Wordlength

$= \Sigma P(v_i).Lw_{word}(v_i)$

= 0.5+0.375+0.5+

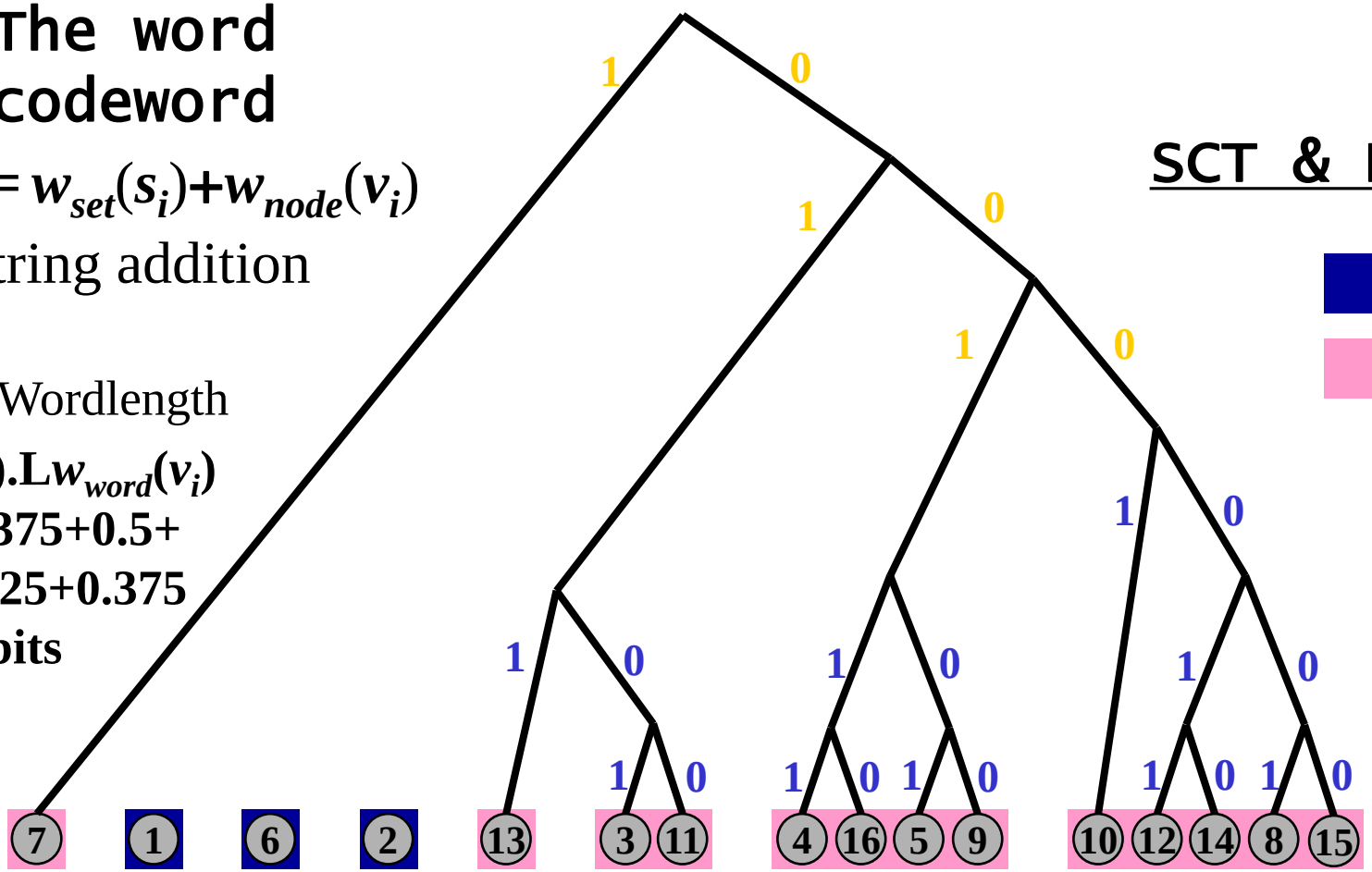
0.625+0.25+0.375

= 2.625 bits

SCT & NCT

{}

≠{}



s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0

node 7: 1
node 13: 011
node 3: 010 1
node 11: 011 0

node 4: 001 11
node 16: 001 10
node 5: 001 01
node 9: 001 00

node 11: 000 1
node 12: 000 011
node 14: 000 010
node 8: 000 001

node 15: 000 000

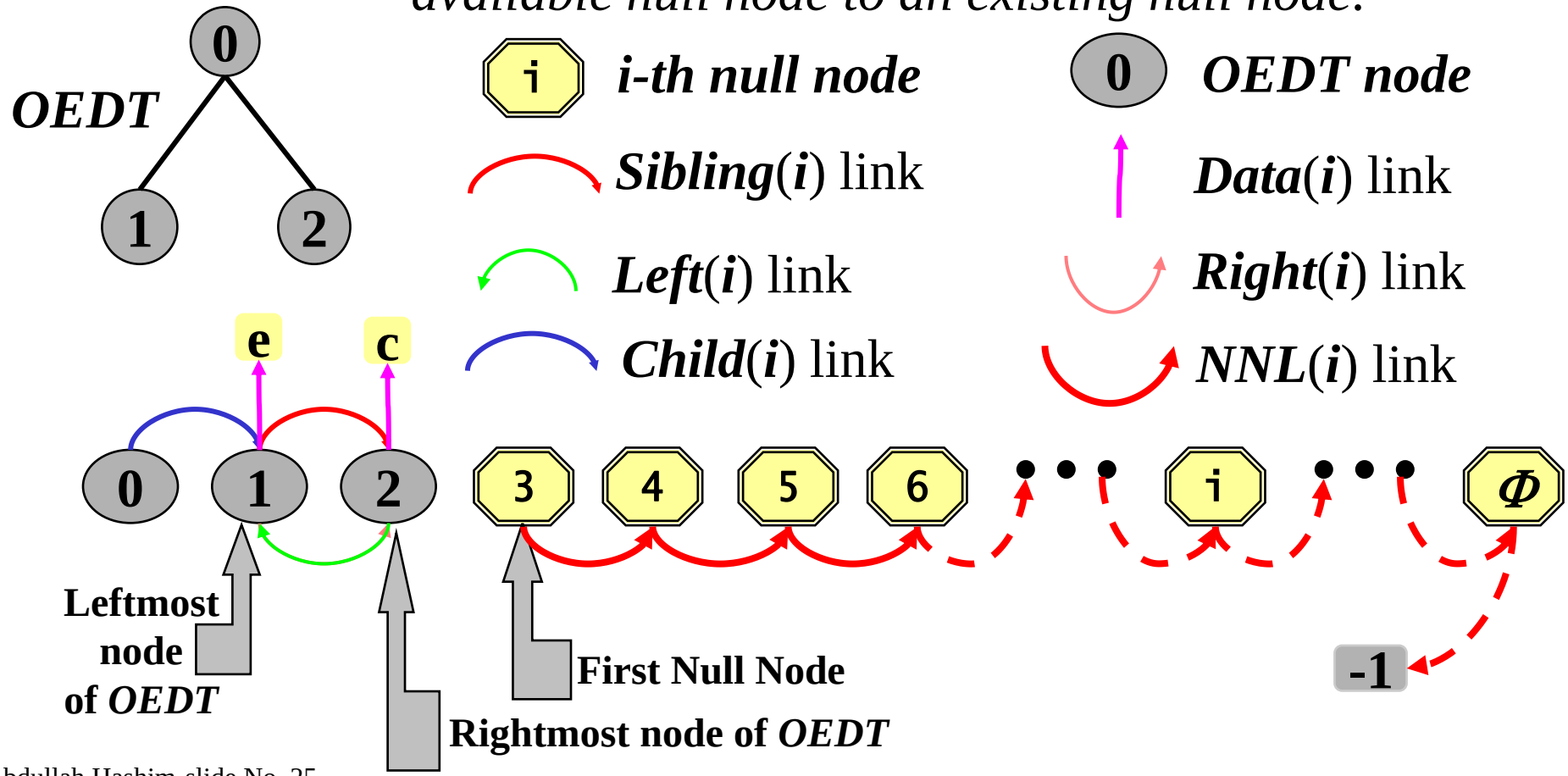
S&M algorithm: The Tree Structure

EDT nodes organisation:

***Null Node*:** *is a node with only null links.*

***Null nodes array*:** *node[i]; i = 0,1,2,.., **Φ**, where **Φ** is the maximum number of words in the **ED**.*

***NNL(i) link*:** *is the **next null node-link** to identify the next available null node to an existing null node.*



S&M algorithm: The Tree Structure

ODT links:

If a link has a value equal -1, the link is a null link.