Title	Choices, choices: Comparing between CHOC'LATE and the classification-tree methodology
Author(s)	Poon, PL; Chen, TY; Tse, TH
Citation	Lecture Notes In Computer Science (Including Subseries Lecture Notes In Artificial Intelligence And Lecture Notes In Bioinformatics), 2012, v. 7308 LNCS, p. 162-176
Issued Date	2012
URL	http://hdl.handle.net/10722/169757
Rights	The original publication is available at www.springerlink.com

Postprint of article in *Reliable Software Technologies: Ada-Europe 2012*,
M. Brorsson and L.M. Pinho (eds.), Lecture Notes in Computer Science,
Springer, Berlin, Germany (2012)

Choices, Choices: Comparing between CHOC'LATE and the Classification-Tree Methodology ⋆

Pak-Lok Poon[1] ⋆⋆, Tsong Yueh Chen[2], and T. H. Tse[3]

[1] School of Accounting and Finance, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
afplpoon@inet.polyu.edu.hk

[2] Faculty of Information and Communication Technologies, Swinburne University of Technology, Hawthorn 3122, Australia
tychen@swin.edu.au

[3] Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong
thtse@cs.hku.hk

Abstract. Two popular specification-based test case generation methods are the choice relation framework and the classification-tree methodology. Both of them come with associated tools and have been used in different applications with success. Since both methods are based on the idea of partition testing, they are similar in many aspects. Because of their similarities, software testers often find it difficult to decide which method to be used in a given testing scenario. This paper aims to provide a solution by first contrasting the strengths and weaknesses of both methods, followed by suggesting practical selection guidelines to cater for different testing scenarios.

Keywords: Choice relation framework, classification-tree methodology, software testing

⋆⋆ Contact author.

1 Introduction

The set of test cases used in software testing, usually known as a *test suite*, should be comprehensive and effective so that any software failure can be revealed [8]. Thus, test suite generation remains a core issue in testing [21]. In general, test cases can be generated according to the program code or the specification. The former approach is known as *code-based* or *white-box* testing while the latter approach is known as *specification-based* or *black-box* testing.

Traditionally, code-based testing received more attention in the literature [9, 15]. In contrast, specification-based testing is relatively less extensively studied, even though its advantages have been widely known [20]. Among various specification-based methods, two popular ones are the choice relation framework and the classification-tree methodology [5–7, 10, 12, 14, 18, 22]. Both of these methods are considered to be useful because they can be applied to *informal* specifications that are primarily written in a narrative language. Both of them come with associated tools. The ***CHOiCe reLATion framEwork*** (***CHOC'LATE***) [7, 18] is an extension of the category-partition method [17] by incorporating formal concepts and practical techniques such as choice relations and their automatic deductions and consistency checks. The ***Classification-Tree Methodology*** (***CTM***) was originally developed by Grochtmann and Grimm [10] and was extended into an integrated classification-tree methodology by Chen et al. [6]. In this paper, for ease of presentation, we will refer to both the (original) classification-tree method and the (extended) integrated classification-tree methodology as CTM.

In general, CHOC'LATE and CTM are input domain partitioning methods [11, 16]. The set of all possible inputs (known as the *input domain*) is divided into subsets (called *subdomains*) according to the specification such that all the elements in each subdomain have essentially the same type of behavior. Test cases are then selected from each subdomain instead of from the entire input domain. In this way, the resulting test suite may better represent the behavior of the software under test.

Despite the growing popularity of CHOC'LATE and CTM, software testers often find it difficult to decide which of them should be used in a given testing scenario, partly because of the similarities among both methods as explained above. This paper aims to provide a solution by first contrasting the strengths and weaknesses of the two methods, followed by suggesting practical selection guidelines to cater for different testing scenarios.

The rest of the paper is structured as follows: Section 2 gives an overview of CHOC'LATE and CTM, and discusses their applicability. Section 3 contrasts the strengths and weaknesses of the two methods in several important aspects. Section 4 then provides guidelines to help the tester decide whether CHOC'LATE or CTM should be used in a given testing scenario. Section 5 discusses some work related to CHOC'LATE and CTM. Finally, Section 6 summarizes the paper.

2 Overview of CHOC'LATE and CTM

2.1 CHOC'LATE

First, let us outline a few fundamental concepts for the understanding of CHOC'LATE [7, 18]. A parameter is an explicit input to a system, while an environment condition is a state of the system. A *category* is a property specified in a parameter or an environment condition that affects the execution behavior of the software under test. For an admission system for a master degree program in accounting, an example of a category is the GMAT score. The possible values associated with a category are partitioned into disjoint subsets known as *choices*. An example of a choice is the set of GMAT scores below 650. Given a category P, P_x is used to denote a choice in P. When there is no ambiguity, we will simply write P_x as x.

A test frame is a set of choices. For instance, a test frame for the qualifications of a master degree applicant is $\{\text{Qualified Accountant}_{\text{yes}}, \text{GMAT Score}_{<650}\}$. A test frame is said to be *complete* if, when an element is selected from every choice in that test frame, a standalone test case can be formed. Suppose the admission system for the master degree program in accounting requires all applicants to state whether they are qualified accountants. Then, $\{\text{Qualified Accountant}_{\text{yes}}, \text{GMAT Score}_{<650}\}$ is a complete test frame but $\{\text{GMAT Score}_{<650}\}$ is incomplete.

Given any choice x, its *relation* with another choice y (denoted by $x \mapsto y$) must be one of the following: (a) x is *fully embedded* in y (denoted by $x \sqsubset y$) if and only if every complete test frame that contains x also contains y. (b) x is *partially embedded* in y (denoted by $x \boxminus y$) if and only if there are some complete test frames that contain both x and y while there are also others that contain x but not y. (c) x is *not embedded* in y (denoted by $x \not\sqsubset y$) if and only if there is no complete test frame that contains both x and y. These three types of choice relations are exhaustive and mutually exclusive, and hence $x \mapsto y$ can be uniquely determined [5, 7, 18].

CHOC'LATE generates a test suite using the following procedure:

(1) Decompose the specification into individual *functional units* that can be tested separately.
(2) Define the categories according to the specification of each functional unit. Partition each category into choices.
(3) Construct a *choice relation table* that captures the constraint (formally known as the *choice relation*) between every pair of choices.
(4) Specify the *preferred maximum number of test frames* $\overline{M}$ and the *minimal priority level* $\underline{m}$. Construct a *choice priority table* that captures the *relative priority levels* (denoted by $r(x)$) of individual choices x. The *lower* the value of $r(x)$, the *higher* will be the priority for x to be used for test frame generation. Any choice x with $r(x) \leqslant \underline{m}$ will always be selected for inclusion as part of a test frame, no matter whether the number of generated test frames exceeds $\overline{M}$.
(5) There are two associated algorithms in CHOC'LATE: one for constructing test frames and the other for extending them. Use the algorithms to generate complete test frames. Form test cases from the complete test frames.

Table 1. Categories and choices for ADMIT.

Categories	Associated Choices
Qualified Accountant	Qualified Accountant$_{\text{yes}}$, Qualified Accountant$_{\text{no}}$
Professional Qualification	Professional Qualification$_{\text{local}}$, Professional Qualification$_{\text{overseas}}$
GMAT Score	GMAT Score$_{<650}$, GMAT Score$_{\geqslant 650}$

Example 1 (Test Suite Generation by CHOC'LATE)
The following is a university admission system (ADMIT) for a master degree program in accounting:

ADMIT captures the following types of information about an applicant in order to determine their eligibility for the program: (a) whether the applicant is a *qualified accountant*, that is, holder of a professional accounting qualification such as CPA; (b) if yes, whether the professional qualification is obtained locally or overseas; and (c) the GMAT score if known. Preference will be given to applicants with a professional accounting qualification, particularly obtained locally. To cater for the situation that an applicant is about to take or has just sat for the GMAT examination, ADMIT allows an applicant to apply for the program before knowing the GMAT score. However, if such an applicant is given a provisional offer, a GMAT score of 650 or above must be obtained before the program starts.

We describe how CHOC'LATE generates a test suite $TS_{\text{ADMIT}}(\text{CHOC})$ for ADMIT:

(1) Because of the simplicity of ADMIT, the specification can be treated as one functional unit in its entirety. No decomposition is needed.
(2) The categories and choices are defined according to ADMIT and shown in Table 1.
(3) The choice relation between every pair of choices is determined according to ADMIT, as shown in the choice relation table $\mathcal{T}_{\text{ADMIT}}$ in Table 2. For example, we have (Professional Qualification$_{\text{local}}$) $\sqsubset$ (Qualified Accountant$_{\text{yes}}$), indicating that every complete test frame containing "Professional Qualification$_{\text{local}}$" must also contain "Qualified Accountant$_{\text{yes}}$". The rationale is that "Professional Qualification$_{\text{local}}$" assumes that the applicant must be a qualified accountant. An example of a partial embedding relation is (Professional Qualification$_{\text{overseas}}$) $\boxed{\sqsubset}$ (GMAT Score$_{\geqslant 650}$). Any complete test frame containing "Professional Qualification$_{\text{overseas}}$" may or may not contain "GMAT Score$_{\geqslant 650}$". This is because a complete test frame containing "Professional Qualification$_{\text{overseas}}$" may contain "GMAT Score$_{<650}$" instead of "GMAT Score$_{\geqslant 650}$", or the complete test frame may not contain any choice from the category "GMAT Score" (when the applicant does not know the GMAT score when applying for the program). Finally, an example of the nonembedding relation is (Qualified Accountant$_{\text{no}}$) $\not\boxed{\sqsubset}$ (Professional Qualification$_{\text{local}}$). In other words, if the applicant is not a qualified accountant, whether the professional accounting qualification is obtained locally or from overseas is irrelevant.

    In essence, the choice relations in $\mathcal{T}_{\text{ADMIT}}$ determine how choices are combined to form complete test frames.
(4) Suppose, for instance, the software tester judges that, according to experience in the application domain, "GMAT Score$_{<650}$" and "GMAT Score$_{\geqslant 650}$" are most likely

Table 2. Choice relation table $\mathcal{T}_{\text{ADMIT}}$ for ADMIT.

	Qualified Accountant$_{\text{yes}}$	**Qualified Accountant**$_{\text{no}}$	**Professional Qualification**$_{\text{local}}$	**Professional Qualification**$_{\text{overseas}}$	**GMAT Score**$_{<650}$	**GMAT Score**$_{\geqslant 650}$
Qualified Accountant$_{\text{yes}}$	$\sqsubset$	$\not\square$	$\mathbb{P}$	$\mathbb{P}$	$\mathbb{P}$	$\mathbb{P}$
Qualified Accountant$_{\text{no}}$	$\not\square$	$\sqsubset$	$\not\square$	$\not\square$	$\mathbb{P}$	$\mathbb{P}$
Professional Qualification$_{\text{local}}$	$\sqsubset$	$\not\square$	$\sqsubset$	$\not\square$	$\mathbb{P}$	$\mathbb{P}$
Professional Qualification$_{\text{overseas}}$	$\sqsubset$	$\not\square$	$\not\square$	$\sqsubset$	$\mathbb{P}$	$\mathbb{P}$
GMAT Score$_{<650}$	$\mathbb{P}$	$\mathbb{P}$	$\mathbb{P}$	$\mathbb{P}$	$\sqsubset$	$\not\square$
GMAT Score$_{\geqslant 650}$	$\mathbb{P}$	$\mathbb{P}$	$\mathbb{P}$	$\mathbb{P}$	$\not\square$	$\sqsubset$

Table 3. Complete test frames generated by CHOC'LATE for ADMIT.

Complete Test Frames
$B_1^c = \{\text{Qualified Accountant}_{\text{yes}}, \text{Professional Qualification}_{\text{local}}\}$
$B_2^c = \{\text{Qualified Accountant}_{\text{yes}}, \text{Professional Qualification}_{\text{local}}, \text{GMAT Score}_{<650}\}$
$B_3^c = \{\text{Qualified Accountant}_{\text{yes}}, \text{Professional Qualification}_{\text{local}}, \text{GMAT Score}_{\geqslant 650}\}$
$B_4^c = \{\text{Qualified Accountant}_{\text{yes}}, \text{Professional Qualification}_{\text{overseas}}\}$
$B_5^c = \{\text{Qualified Accountant}_{\text{yes}}, \text{Professional Qualification}_{\text{overseas}}, \text{GMAT Score}_{<650}\}$
$B_6^c = \{\text{Qualified Accountant}_{\text{yes}}, \text{Professional Qualification}_{\text{overseas}}, \text{GMAT Score}_{\geqslant 650}\}$
$B_7^c = \{\text{Qualified Accountant}_{\text{no}}\}$
$B_8^c = \{\text{Qualified Accountant}_{\text{no}}, \text{GMAT Score}_{<650}\}$
$B_9^c = \{\text{Qualified Accountant}_{\text{no}}, \text{GMAT Score}_{\geqslant 650}\}$

to reveal faults in ADMIT. In this case, the tester will assign higher priorities to $r(\text{GMAT Score}_{<650})$ and $r(\text{GMAT Score}_{\geqslant 650})$ than other choices. As a result, "GMAT Score$_{<650}$" and "GMAT Score$_{\geqslant 650}$" will first be used to generate test frames. Suppose further that, after considering the testing resources available, the tester sets $\overline{M}$ to a very high value, indicating to the associated algorithms that all complete test frames are to be generated for testing. (See Section 3.4 for more details.)

(5) The associated algorithms generate a set of test frames $SF_{\text{ADMIT}}(\text{CHOC})$ for ADMIT. In particular, nine test frames are complete, as shown in Table 3. Obviously, the test frames B_2^c, B_3^c, B_5^c, B_6^c, B_8^c, and B_9^c are complete. The test frames B_1^c, B_4^c, and B_7^c are also complete because ADMIT allows an applicant to apply for the program before knowing the GMAT score, so that the score is not a necessary input to the system.

During the generation process, several incomplete test frames are also formed. An example is $\{\text{Qualified Accountant}_{\text{yes}}, \text{GMAT Score}_{<650}\}$. It is incomplete because it needs a choice from the category "Professional Qualification" to form a complete test frame. The tester needs to check $SF_{\text{ADMIT}}(\text{CHOC})$ and remove any incomplete test frames. After the removal process, the nine complete test frames in Table 3 remain in $SF_{\text{ADMIT}}(\text{CHOC})$. For each of these complete test frames, a test case is formed by randomly selecting and combining an element from each choice in that test frame. Consider, for instance, B_9^c. A test case $\{\text{Qualified Accountant} = \text{no}, \text{GMAT Score} = 720\}$ can be formed. ■

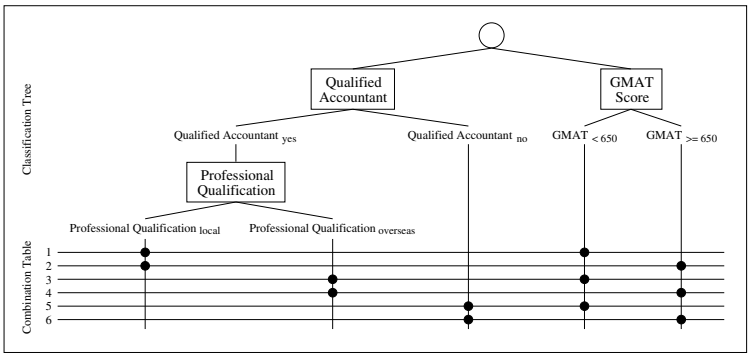

Fig. 1. Classification tree Υ_{ADMIT} and combination table for ADMIT

2.2 CTM

CTM is similar to CHOC'LATE in its approach to test suite generation [6, 10, 12, 22]. It consists of the following steps:

(1) Decompose the specification into individual functional units that can be tested separately. This step is identical to step (1) in CHOC'LATE.
(2) For each functional unit, identify *classifications* and their associated *classes*. Classifications and classes in CTM are identical to categories and choices, respectively, in CHOC'LATE. For ease of presentation, in the rest of this paper, classifications and classes will be stated as categories and choices.
(3) Construct a *classification tree* to capture the relation between any choice P_x and any category Q ($\neq P$).
(4) Use the associated algorithm to construct the *combination table*, through which valid combinations of choices are selected as complete test frames. A test case is then formed from each complete test frame as in CHOC'LATE.

Example 2 (Test Suite Generation by CTM)
Refer to the university admission system ADMIT in Example 1. Steps (1) and (2) of CTM are identical to their counterparts in CHOC'LATE. Let us illustrate steps (3) and (4) of CTM for generating a set of test frames $SF_{\text{ADMIT}}(\text{CTM})$ and its corresponding test suite $TS_{\text{ADMIT}}(\text{CTM})$:

(3) We construct a classification tree Υ_{ADMIT} (as shown in the upper half of Fig. 1), capturing the relations between the relevant categories and choices. Categories in the classification tree are enclosed in boxes whereas choices are not.

A small circle at the top of a classification tree is the *general root node*, covering the entire input domain. The categories directly under the general root node, such as "Qualified Accountant" and "GMAT Score" in Fig. 1, are called *top-level categories*. In general, a category P may have several choices P_x directly under it. P is known as the *parent category* and P_x is known as a *child choice*. In Fig. 1, for

example, "Qualified Accountant" is the parent category of "Qualified Accountant$_{\text{yes}}$" whereas "Qualified Accountant$_{\text{yes}}$" is a child choice of "Qualified Accountant". Similarly, a choice P_x may have one or more categories Q ($\neq P$) directly under it. Then P_x is known as the *parent choice* and Q is known as a *child category*. In Fig 1, for example, "Qualified Accountant$_{\text{yes}}$" is the parent choice of "Professional Qualification" while "Professional Qualification" is the child category of "Qualified Accountant$_{\text{yes}}$".

(4) Use the associated algorithm to construct the combination table and to generate complete test frames (as shown, for example, in the lower half of Fig. 1). The process makes use of the following rules:

   (a) Draw the grids of the combination table under a classification tree. The columns of the table correspond to the terminal nodes of the classification tree. The rows correspond to test frames.
   (b) Generate a test frame in the combination table by selecting a combination of choices in a classification tree as follows: (i) select one and only one child choice for each top-level category, and (ii) for every child category of each selected choice, recursively select one and only one child choice.

   For the given classification tree Υ_{ADMIT}, the above rules generate SF_{ADMIT}(CTM) containing six test frames. For instance, the test frame corresponds to row 1 of the combination table is {Qualified Accountant$_{\text{yes}}$, Professional Qualification$_{\text{local}}$, GMAT Score$_{<650}$}. Since a classification tree may not fully capture the relations among the relevant categories and choices, resulting in the occurrence of incomplete test frames, we need to check the set of test frames generated and remove any incomplete ones. After checking, we find that all the six test frames in SF_{ADMIT} (CTM) are complete. For each of these test frames, a test case is formed by randomly selecting and combining an element from each choice in that test frame. ■

2.3 Applicability of CHOC'LATE and CTM

It is obvious that a testing method may not be applicable to all types of systems. CHOC'LATE and CTM are no exception. Both methods are not specifically developed for testing real-time systems or embedded systems. Having said that, it should be noted that CHOC'LATE and CTM are generic testing methods and, as such, they can be used to generate test suites when the following two conditions are met: (a) the software can be decomposed into functional units to be tested independently, and (b) categories, choices, and relations at the category-level or at the choice-level can be identified from the specification. For example, CHOC'LATE has been successfully applied to different application domains, including the inventory registration module and the purchase-order generation module of an inventory management system, an online telephone inquiry system, and the meal scheduling module of an airline meal ordering system [7]. As for CTM, its successful applications to an airfield lighting control system, an automatic mail sorting system, an integrated ship management system, and a parser as part of a software development environment have been reported [10].

3 Strengths and Weaknesses of CHOC'LATE and CTM

3.1 Relations among Categories and Choices

CHOC'LATE and CTM use different approaches to capture and represent relations among choices or categories. These relations then determine how choices are combined together to form complete test frames. CHOC'LATE captures the relation between every pair of *choices*. They are expressed in terms of three choice relations (full embedding, partial embedding, and nonembedding) and captured in a choice relation table. In contrast, CTM captures the relations at the *category* level, or more specifically, the relations between a choice P_x and a category Q ($\neq P$). Furthermore, these relations are expressed in a hierarchical tree structure known as a classification tree. Obviously, category-level constraints are coarser than choice-level constraints. On the other hand, since the number of category-level constraints is much less than that of choice-level constraints, the former type requires less effort to identify.

Consider, for example, the classification tree Υ_{ADMIT} in Fig. 1. According to the selection rules, because "Qualified Accountant$_{\text{yes}}$" is the parent choice of "Professional Qualification", whenever either "Professional Qualification$_{\text{local}}$" or "Professional Qualification$_{\text{overseas}}$" is selected to form part of any complete test frame, "Qualified Accountant$_{\text{yes}}$" must also be selected. This part of the tree structure is similar in effect to the definition of the choice relations (Professional Qualification$_{\text{local}}$ $\sqsubset$ Qualified Accountant$_{\text{yes}}$) and (Professional Qualification$_{\text{overseas}}$ $\sqsubset$ Qualified Accountant$_{\text{yes}}$) in CHOC'LATE.

Because CHOC'LATE captures the relations at a more fine-grained level (namely, the choice level instead of the category level), CHOC'LATE is generally more comprehensive in generating complete test frames. Let us compare SF_{ADMIT}(CHOC) and SF_{ADMIT}(CTM). As explained in Example 1, SF_{ADMIT}(CHOC) contains all the nine complete test frames $B_1^c, B_2^c, \ldots, B_9^c$ that should be generated, as shown in Table 3. SF_{ADMIT}(CTM), however, only contains six complete test frames, namely B_2^c, B_3^c, B_5^c, B_6^c, B_8^c, and B_9^c, corresponding to rows 1, 2, ..., 6 in the combination table of Fig. 1 (see Example 2). In other words, CTM cannot generate the complete test frames B_1^c, B_4^c, and B_7^c. This problem affects the comprehensiveness of SF_{ADMIT}(CTM) and TS_{ADMIT}(CTM), and hence the effectiveness of testing.

A close examination of the structure of the classification tree Υ_{ADMIT} in Fig. 1 reveals the reason for the omission of B_1^c, B_4^c, and B_7^c. "GMAT Score" is a top-level category in Υ_{ADMIT}. According to the selection rules, a child choice of "GMAT Score" must be selected as part of any complete test frame. This requirement prevents B_1^c, B_4^c, and B_7^c from being generated, because all these three complete test frames do not contain any choice in "GMAT Score". [4]

In contrast, CHOC'LATE can generate B_1^c, B_4^c, and B_7^c by using the partial embedding relation. For example, by defining (Qualified Accountant$_{\text{yes}}$ $\sqsubset\!\!\!\!\cdot$ GMAT Score$_{<650}$), (Qualified Accountant$_{\text{yes}}$ $\sqsubset\!\!\!\!\cdot$ GMAT Score$_{\geqslant 650}$) and other relevant choice relations (see

[4] One may argue that Υ_{ADMIT} is only one of the many possible tree structures with respect to the categories and choices in Table 1. We must point out, however, that no matter how a classification tree is drawn using these categories and choices, it is unable to generate all the nine complete test frames in Table 3.

Table 2), any complete test frame B^c generated by CHOC'LATE containing "Qualified Accountant$_{\text{yes}}$" must be one of the following three types:

(a) B^c contains "GMAT Score$_{<650}$" but does not contain "GMAT Score$_{\geqslant 650}$",
(b) B^c contains "GMAT Score$_{\geqslant 650}$" but does not contain "GMAT Score$_{<650}$", and
(c) B^c does not contain both "GMAT Score$_{<650}$" and "GMAT Score$_{\geqslant 650}$".

Because of type (c), B^c_1 and B^c_4 (which are omitted from $SF_{\text{ADMIT}}(\text{CTM})$) will exist in $SF_{\text{ADMIT}}(\text{CHOC})$. Similarly, we can define (Qualified Accountant$_{\text{no}}$ $\boxed{\text{E}}$ GMAT Score$_{<650}$), (Qualified Accountant$_{\text{no}}$ $\boxed{\text{E}}$ GMAT Score$_{\geqslant 650}$), and other relevant choice relations to guarantee the generation of B^c_7.

3.2 Inherent Limitation of Tree Structure

Given any pair of distinct categories P and Q, Chen et al. [6] define four possible types of hierarchical relations: (a) P is a *loose ancestor* of Q (denoted by $P \Leftrightarrow Q$), (b) P is a *strict ancestor* of Q (denoted by $P \Rightarrow Q$), (c) P is *incompatible with* Q (denoted by $P \sim Q$), and (d) P has *other relations with* Q (denoted by $P \otimes Q$). Note that, for the ancestor relation, type (a) is symmetric whereas type (b) is anti-symmetric. Readers may refer to [6] for details.

The hierarchical relations (b), (c), and (d) affect the relative positions of P and Q in a classification tree. Consider, for example, the categories "Qualified Accountant" and "Professional Qualification" in the classification tree Υ_{ADMIT}. We have (Qualified Accountant $\Rightarrow$ Professional Qualification), causing "Professional Qualification" to appear under the choice "Qualified Accountant$_{\text{yes}}$" (but not "Qualified Accountant$_{\text{no}}$") of "Qualified Accountant".

On the other hand, for relation (a), it indicates a *symmetric* parent-child or ancestor-descendent hierarchical relation between P and Q, resulting in a loop in a classification tree. This relation violates an implicit assumption of classification trees, namely, that the parent-child or ancestor-descendent hierarchical relation must be *anti-symmetric* for any pair of categories; otherwise a classification tree cannot be constructed. Since CHOC'LATE does not use a tree structure to capture the relations among choices, the problem associated with the loose-ancestor relation is not applicable.

Example 3 (Loose-Ancestor Hierarchical Relation)
Suppose we have a pair of distinct categories P and Q, where P has two associated choices P_x and P_y, and Q has two associated choices Q_a and Q_b. Suppose further that, with respect to P and Q, only three complete test frames exist, namely, $B^c_1 = \{P_x, Q_a\}$, $B^c_2 = \{P_y\}$, and $B^c_3 = \{Q_b\}$. In view of these complete test frames and according to the definitions of hierarchical relations [6], we have a loose ancestor relation $P \Leftrightarrow Q$. Thus, a classification tree cannot be constructed to *fully* capture the relations between P and Q such that B^c_1, B^c_2, and B^c_3 can be generated.

If we ignore the ability of classification trees in generating all complete test frames, various tree structures can be constructed from P, Q, and their associated choices, including the three depicted in Fig. 2.[5] None of these classification trees (including

[5] There are other feasible classification trees that are not shown in the figure, because a category and its associated choices may occur more than once in a classification tree [6].

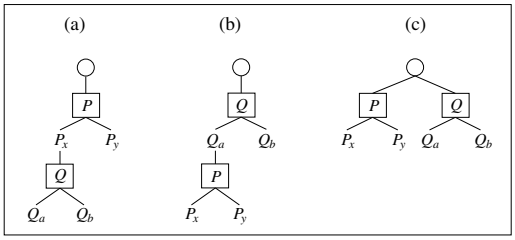

Fig. 2. Three possible classification trees

the three in Fig. 2 and others not included in the figure) generates all the complete test frames B_1^c, B_2^2, and B_3^c. Consider, for instance, the tree in Fig. 2(a). It generates B_1^c and B_2^c but not B_3^c.

In CHOC'LATE, we can define the following choice relations between P and Q: (i) $P_x \sqsubset Q_a$ and $Q_a \sqsubset P_x$; (ii) $P_y \not\sqsupset m$ and $m \not\sqsupset P_y$, where choice $m = P_x$, Q_a, or Q_b; and (iii) $Q_b \not\sqsupset n$ and $n \not\sqsupset Q_b$, where choice $n = P_x$, P_y, or Q_a. These definitions will then cause the associated algorithms to generate B_1^c, B_2^c, and B_3^c, respectively. ■

3.3 Automatic Deduction and Consistency Checking of Relations

The comprehensiveness of the generated test suite depends on the correctness of choice relations and hierarchical relations in CHOC'LATE and CTM, respectively. However, it would be tedious and error prone to manually define all such relations. Chen et al. [7] have identified various properties of these relations in CHOC'LATE to form the basis for their automatic deductions and consistency checking. Two examples are: **(Property 1)** Given any choices x, y, and z, if $x \sqsubset y$ and $x \boxplus z$, then $y \boxplus z$. **(Property 2)** Given any choices x, y, and z, if $x \sqsubset z$ and $y \boxplus z$, then $y \boxplus x$ or $y \not\sqsupset x$.

Property 1 provides a basis for automatic deduction of choice relations because its "then" part consists of a definite relation. Thus, once $x \sqsubset y$ and $x \boxplus z$ are defined, $y \boxplus z$ can be automatically deduced without manual intervention. As for Property 2, its "then" part contains two possible relations. Although the property cannot be used for automatic deduction, it nevertheless allows us to check the consistency of the relations among choices. For example, we know that when $x \sqsubset z$ and $y \boxplus z$, we cannot have $y \sqsubset x$, or else it will contradict Property 2.

Similar properties and techniques have been identified in CTM [6]. Two examples are: **(Property 3)** Given any categories P and Q, if $P \Rightarrow Q$, then $Q \otimes P$. **(Property 4)** Given any categories P and Q, if $P \otimes Q$, then $Q \Rightarrow P$ or $Q \otimes P$. Properties 3 and 4 can be used for automatic deduction and consistency checking of hierarchical relations, respectively.

The techniques of automatic deduction and consistency checking are more advanced and refined in CHOC'LATE than in CTM. In CHOC'LATE, there are five main propositions and three main corollaries, from which properties such as those mentioned above are derived. Some of these main propositions and corollaries are further refined into sub-propositions and sub-corollaries (see [7] for details). On the other hand, in CTM [6], only three propositions exist and they cannot be further refined.

3.4 Test Frame Generation

Often, many categories and choices can be defined from a real-life specification [3]. Consequently, CHOC'LATE and CTM will generate many complete test frames (and hence many test cases) to cover diverse valid combinations of the defined choices. For instance, it has been reported that real-world protocol software may have 448–2402 test cases per test suite [13]. Such a test suite can be prohibitively expensive to execute exhaustively owing to its large size.

To alleviate this problem, CHOC'LATE allows testers to control the total number of test frames generated by specifying (a) the *preferred* maximum number of test frames $\overline{M}$, (b) the relative priority level $r(x)$ of each individual choice x, and (c) the minimal priority level $\underline{m}$. For $\overline{M}$, the word "preferred" implies that the limit is not absolute, as it may be overwritten by $\underline{m}$. For the relative priority level of individual choices, they determine the *order* of choices used for test frame generation. The lower the value of $r(x)$, the higher will be the priority of x. $\underline{m}$ allows testers to ensure that those choices x with $r(x) \leqslant \underline{m}$ will always be selected for inclusion as part of a test frame, no matter whether the number of generated test frames exceeds $\overline{M}$ or not. In the situation where $\overline{M}$ should not be waived by $\underline{m}$, $\underline{m}$ should be set to zero, and $\overline{M}$ becomes the *absolute* maximum number of generated test frames.

Testers often face a dilemma that, on one hand, they prefer to set a maximum number $\overline{M}$ of generated test frames so as to control the testing effort, but on the other hand, the choices considered very important should always be used for test frame generation, even though this may cause the number of generated test frames to exceed $\overline{M}$. Allowing testers to set the values of $\overline{M}$, $\underline{m}$, and the relative priority level of choices will provide them with flexibility in dealing with such dilemma.

In contrast, CTM aims at generating valid combinations of choices as complete test frames without considering the testing resources involved. Grochtmann and Grimm [10] argue that maximality and minimality criteria can be incorporated into CTM, thus allowing testers to control the number of complete test frames to some extent. The *maximality* criterion naturally requires each valid combination of choices to form a complete test frame. The *minimality* criterion, on the other hand, requires each choice to be used in at least one complete test frame, so that the number of complete test frames can be reduced. Obviously, even with these two criteria, the ability to control the number of generated complete test frames in CTM is far more restricted than when compared with CHOC'LATE.

3.5 Documentation of the Software under Test

Both CHOC'LATE and CTM aim to generate a test suite for software testing. In addition, during the generation process, the choice relation table constructed in CHOC'LATE and the classification tree constructed in CTM can serve as useful documentation of the software under test [10].

Briand et al. [2] argue that "devising ... categories and choices is ... necessary to understand the rationale behind test cases and is a way for the tester to formalize her understanding of the functional specification". In CTM, a classification tree is constructed, capturing the relations among relevant categories and choices. Since it is in

a graphic form, a classification tree is more concise and descriptive than a narrative specification [10]. Similarly, a two-dimensional choice relation table in CHOC'LATE captures the relation between every pair of choices. This table is better than a narrative specification for the purpose of documentation and reasoning.

When comparing a classification tree with a choice relation table, there are mixed opinions. On one hand, some people prefer a classification tree to a choice relation table for the purpose of *presentation*. They argue that the pictorial simplicity and vividness of a tree makes it more understandable [23]. On the other hand, others argue that a choice relation table is better than a classification tree because the former contains more fine-grained information to help readers understand the relations among individual choices (see Section 3.1 for details).

4 Selection Guidelines

Intuitively, every testing method has its own merits and drawbacks. CHOC'LATE and CTM are no exception. Neither of them is ideal for every testing situation. A software tester should be knowledgeable enough to decide whether CHOC'LATE or CTM is best applied to specific testing scenarios. The decision is not straightforward because both of them are input domain partitioning methods [11, 16] and hence they are fairly similar. We provide below some guidelines to help a tester decide which of them should be used in a given testing scenario.

Given a specification, the tester should first consider the level of abstraction of the constraints and the relationships among constraints. If the constraints are specified at the choice level and the tester can afford the effort to identify their relationships, then CHOC'LATE is preferred because it will generate a more comprehensive set of complete test frames (see Section 3.1). On the other hand, if all or most of the constraints are only available at the category level, or if the tester can only afford to identify category-level relationships among constraints, or if the tester prefers an intuitive graphic presentation of the relations among constraints (see Section 3.5), then CTM is the option (see also Section 3.1).

In addition, the possible occurrence of the loose-ancestor hierarchical relation ($P \Leftrightarrow Q$) between two distinct categories P and Q is another factor to consider. If this relation exists, then CTM should not be chosen (see Section 3.2), unless the use of CHOC'LATE is prohibited by other factors such as the absence of choice-level constraints in the specification as explained above.

Next, we consider the process of generating complete test frames. Ideally, the process must be well executed so that no complete test frame will be missing. Otherwise, testing may not be comprehensive and some software failures may never be revealed. In the generation process, the correctness of the constraints (at the category or choice level) is of utmost importance because it will affect the comprehensiveness of the set of complete test frames generated. If the number of constraints to be manually defined is large (especially when the specification is large and complex), the chance of making mistakes is high. In this regard, the complexity of the choice relation table in CHOC'LATE is an additional consideration that needs to be taken into account when selecting between the two methods. In any case, both CHOC'LATE and CTM offer the

features of automatic deduction and consistency checking of relations, with a view to improving the effectiveness and efficiency of constraint definitions. The two features provided by CHOC'LATE are more advanced and refined than those by CTM. This may serve to counterbalance the complexity of the choice relation table in CHOC'LATE (see Section 3.3).

The amount of testing resources available is also an important factor. As we have mentioned, it would be ideal to test the software with all the complete test frames. In reality, however, this may be infeasible because of the shortage of testing resources. If this happens, both CHOC'LATE and CTM allow the tester to select a subset of all complete test frames to be generated for testing. Among the two methods, CHOC'LATE is more refined in allowing the tester to control how this subset is generated. Therefore, if testing constraints are an issue, CHOC'LATE will be a better choice (see Section 3.4).

5 Related Work

Yu et al. [24] proposed some enhancements to CTM by annotating a classification tree with additional information (including selector expressions, occurrence tags, and weight tags) to reduce manual effort in the generation, selection, and prioritization of test cases. They also developed an automated tool (EXTRACT) that implements the proposed enhancements.

Amla and Ammann [1] analyzed the feasibility of applying the category-partition method (on which CHOC'LATE is based) to Z specifications and found that testing requirements can be defined from formal specifications more easily. Hierons, Singh and their co-workers [12, 22] have also done similar work in the context of Z specifications. They introduce an approach [22] to generating test cases from Z specifications by combining CTM with disjunctive normal forms, and present another approach [12] to extracting predicates from Z specifications and building a classification tree from these predicates.

Obviously, the comprehensiveness of a test suite generated by CHOC'LATE and CTM depends on how well categories and choices are identified from the specification. In this regard, Chen, Poon, and their co-workers [4, 19] have conducted several empirical studies to investigate the common mistakes made by experienced and inexperienced testers when the identification process is done in an ad hoc manner. Furthermore, they have recently developed a **D**ivid**E**-and-conquer methodology for identifying categorie**S**, choice**S**, and choic**E** **R**elations for **T**est case generation (**DESSERT**) for large and complex specifications that involve many different components [5].

6 Summary and Conclusion

In this paper, we have outlined the main concepts of two popular specification-based testing methods, namely, CHOC'LATE and CTM. We have used examples to illustrate how both methods generate a test suite from the specification, and contrasted their strengths and weaknesses with respect to five different aspects, namely, (a) relations among categories and choices, (b) inherent limitation of the tree structure, (c) automatic

deduction and consistency checking of relations, (d) test frame generation, and (e) documentation of the software under test. Based on these strengths and weaknesses, we have provided guidelines to help the tester decide which method to use under different testing scenarios. Thus, the paper will help the software testing community better understand CHOC'LATE and CTM, and determine which of them is more appropriate in a specific testing scenario.

Acknowledgment This research is supported in part by a Discovery Grant of the Australian Research Council (project no. ARC DP0771733) and the General Research Fund of the Research Grants Council of Hong Kong (project no. 717811).

References

1. Amla, N., Ammann, P. E.: Using Z Specifications in Category Partition Testing. In: Systems Integrity, Software Safety, and Process Security: Building the Right System Right: Proceedings of the 7th Annual IEEE Conference on Computer Assurance (COMPASS 1992), pp. 3–10. IEEE Computer Society, Los Alamitos, CA (1992)
2. Briand, L. C., Labiche, Y., Bawar, Z., Spido, N. T.: Using Machine Learning to Refine Category-Partition Test Specifications and Test Suites. Information and Software Technology 51 (11), 1551–1564 (2009)
3. Chan, E. Y. K., Chan, W. K., Poon, P.-L., Yu, Y. T.: An Empirical Evaluation of Several Test-a-Few Strategies for Testing Particular Conditions. Software: Practice and Experience (2011) doi: 10.1002/spe.1098
4. Chen, T. Y., Poon, P.-L., Tang, S.-F., Tse, T. H.: On the Identification of Categories and Choices for Specification-Based Test Case Generation. Information and Software Technology 46 (13), 887–898 (2004)
5. Chen, T. Y., Poon, P.-L., Tang, S.-F., Tse, T. H.: DESSERT: a DividE-and-conquer methodology for identifying categorieS, choiceS, and choicE Relations for Test case generation. IEEE Transactions on Software Engineering (2011) doi: 10.1109/TSE.2011.69
6. Chen, T. Y., Poon, P.-L., Tse, T. H.: An Integrated Classification-Tree Methodology for Test Case Generation. International Journal of Software Engineering and Knowledge Engineering 10 (6), 647–679 (2000)
7. Chen, T. Y., Poon, P.-L., Tse, T. H.: A Choice Relation Framework for Supporting Category-Partition Test Case Generation. IEEE Transactions on Software Engineering 29 (7), 577–593 (2003)
8. Chusho, T.: Test Data Selection and Quality Estimation Based on the Concept of Essential Branches for Path Testing. IEEE Transactions on Software Engineering 13 (5), 509–517 (1987)
9. Foreman, L. M., Zweben, S. H.: A Study of the Effectiveness of Control and Data Flow Testing Strategies. Journal of Systems and Software 21 (3), 215–228 (1993)
10. Grochtmann, M., Grimm, K.: Classification Trees for Partition Testing. Software Testing, Verification and Reliability 3 (2), 63–82 (1993)
11. Hierons, R. M., Harman, M., Fox, C., Ouarbya, L., Daoudi, M.: Conditioned Slicing Supports Partition Testing. Software Testing, Verification and Reliability 12 (1), 23–28 (2002)

12. Hierons, R. M., Harman, M., Singh, H.: Automatically Generating Information from a Z Specification to Support the Classification Tree Method. In: Proceedings of the 3rd International Conference of B and Z Users. LNCS, vol. 2651, pp. 388–407. Springer, Berlin, Germany (2003)
13. Jiang, B., Tse, T. H., Grieskamp, W., Kicillof, N., Cao, Y., Li, X., Chan, W. K.: Assuring the Model Evolution of Protocol Software Specifications by Regression Testing Process Improvement. Software: Practice and Experience 41 (10), 1073–1103 (2011)
14. Kansomkeat, S., Thiket, P., Offutt, J.: Generating Test Cases from UML Activity Diagrams Using the Condition-Classification Tree Method. In: Proceedings of the 2nd International Conference on Software Technology and Engineering (ICSTE 2010), pp. V1-62–V1-66. IEEE Computer Society, Los Alamitos, CA (2010)
15. Lemos, O. A. L., Vincenzi, A. M. R., Maldonado, J. C., Masiero, P. C.: Control and Data Flow Structural Testing Criteria for Aspect-Oriented Programs. Journal of Systems and Software 80 (6), 862–882 (2007)
16. Myers, G. J.: The Art of Software Testing. Wiley, Hoboken, NJ (2004)
17. Ostrand, T. J., Balcer, M. J.: The Category-Partition Method for Specifying and Generating Functional Tests. Communications of the ACM 31 (6), 676–686 (1988)
18. Poon, P.-L., Tang, S.-F., Tse, T. H., Chen, T. Y.: CHOC'LATE: a Framework for Specification-Based Testing. Communications of the ACM 53 (4), 113–118 (2010)
19. Poon, P.-L., Tse, T. H., Tang, S.-F., Kuo, F.-C.: Contributions of Tester Experience and a Checklist Guideline to the Identification of Categories and Choices for Software Testing. Software Quality Journal 19 (1), 141–163 (2011)
20. Richardson, D. J., O'Malley, O., Tittle, C.: Approaches to Specification-Based Testing. In: Proceedings of the ACM SIGSOFT 3rd Symposium on Software Testing, Analysis, and Verification (TAV 3), pp. 86–96. ACM, New York, NY (1989)
21. Shepard, T., Lamb, M., Kelly, D.: More Testing should be Taught. Communications of the ACM 44 (6), 103–108 (2001)
22. Singh, H., Conrad, M., Sadeghipour, S.: Test Case Design Based on Z and the Classification-Tree Method. In: Proceedings of the 1st IEEE International Conference on Formal Engineering Methods (ICFEM 1997), pp. 81–90. IEEE Computer Society, Los Alamitos, CA (1997)
23. Subramanian, G. H., Nosek, J., Raghunathan, S. P., Kanitkar, S. S.: A Comparison of the Decision Table and Tree. Communications of the ACM 35 (1), 89–94 (1992)
24. Yu, Y. T., Ng, S. P., Chan, E. Y. K.: Generating, Selecting and Prioritizing Test Cases from Specifications with Tool Support. In: Proceedings of the 3rd International Conference on Quality Software (QSIC 2003), pp. 83–90. IEEE Computer Society, Los Alamitos, CA (2003)