Тестирование методом «черного ящика»

Метод причинно-следственных диаграмм

Метод прогнозирования ошибок

Причинно-следственная диаграмма

- Недостаток методов разбиения данных на классы эквивалентности и анализа граничных значений – отсутствие проверки комбинации входных условий
- Количество возможных комбинаций входных условий очень велико, поэтому требуется систематический подход к выбору комбинаций для тестирования
- Метод причинно-следственных диаграмм (диаграмм Исикавы) позволяет выбрать некоторый набор тестов, а также обнаружить неполноту и неоднозначность исходных спецификаций
- Причинно-следственная диаграмма является упрощенным аналогом цифровой логической схемы
- Недостатком этого подхода является плохое исследование граничных условий

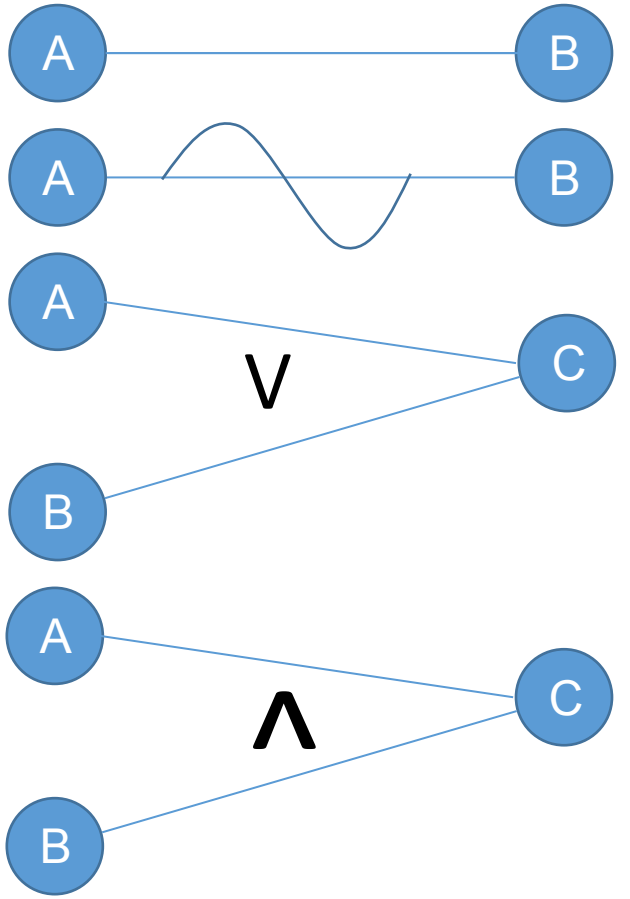
Этапы построения диаграммы

- Спецификация разбивается на отдельные части
- В спецификации определяются множество причин и следствий. Под причиной понимается отдельное входное условие или класс эквивалентности. Следствие представляет собой выходное условие или преобразование системы. Каждой причине и следствию присваивается номер
- На основе анализа семантического (смыслового) содержания спецификации строится булевый граф, связывающий причины и следствия
- Диаграмма преобразуется в таблицу решений с ограниченными входами
- Каждый столбец таблицы решений соответствует тесту
- Диаграмма снабжается примечаниями, задающими ограничения и описывающими комбинации причин и следствий, реализация которых невозможна

Основные элементы диаграммы

Каждый узел диаграммы может находится, либо в состоянии 0 («отсутствует»), либо 1 («присутствует»)

- Тождество (если A=1, то B=1)
- Отрицание (если A=1, то B=0)
- ИЛИ (если A=1 или B=1, то C=1)
- И (если A=1 и B=1, то C=1)

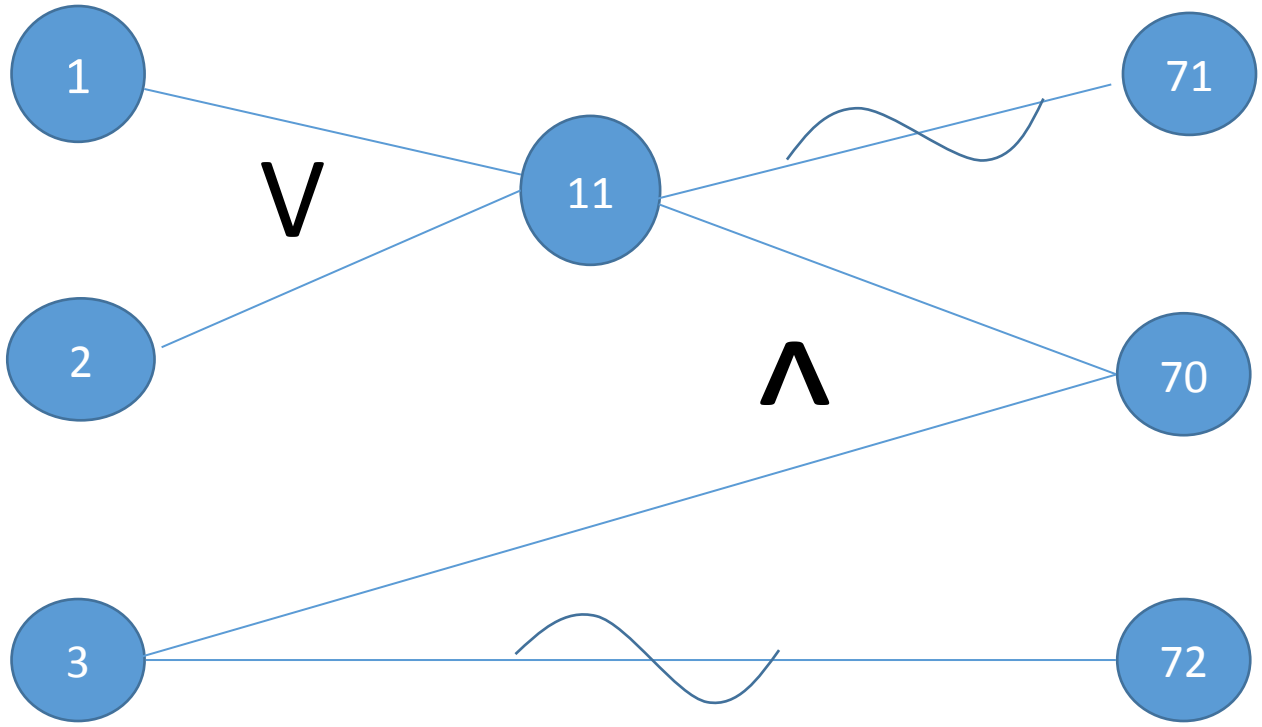


Пример

- Спецификация: символ в колонке 1 должен быть буквой «A» или «B», а в колонке 2 – цифрой. В этом случае файл обновляется. Если первый символ неправильный, то выдается сообщение X12, а если второй символ не цифра – сообщение X13
- Причины:
  - 1 – символ «A» в колонке 1
  - 2 – символ «B» в колонке 1
  - 3 – цифра в колонке 2
- Следствия:
  - 70 – файл обновляется
  - 71 – выдается сообщение X12
  - 72 – выдается сообщение X13

Пример диаграммы

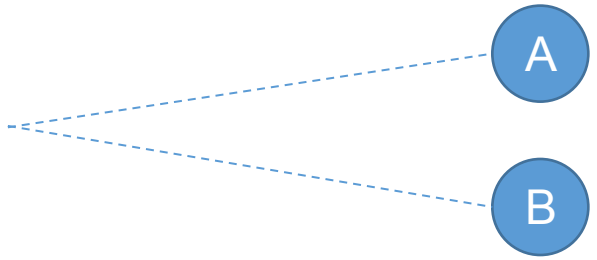
- Символ в колонке 1 должен быть буквой «А» или «В», а в колонке 2 – цифрой. В этом случае файл обновляется
- Если первый символ неправильный, то выдается сообщение X12, а если второй символ не цифра – сообщение X13



Символы ограничений на диаграмме

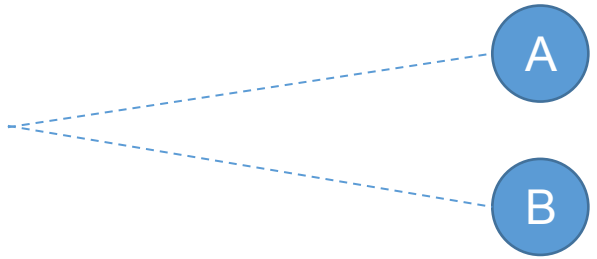
- Только одно значение может быть 1

E



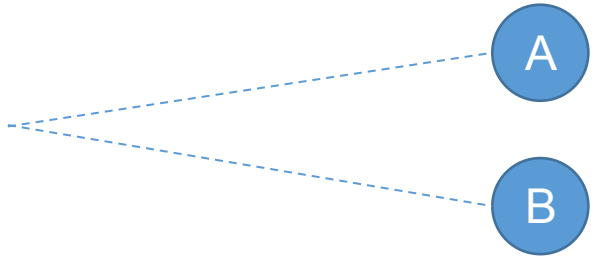
- Одно из значений обязано равняться 1

I



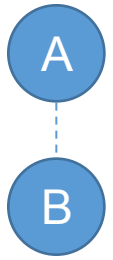
- Одно и только одно из значений обязано равняться 1

O



- A=1 только, если B=1

R



Пример использования ограничений

Первый символ не может быть одновременно «А» и «В»

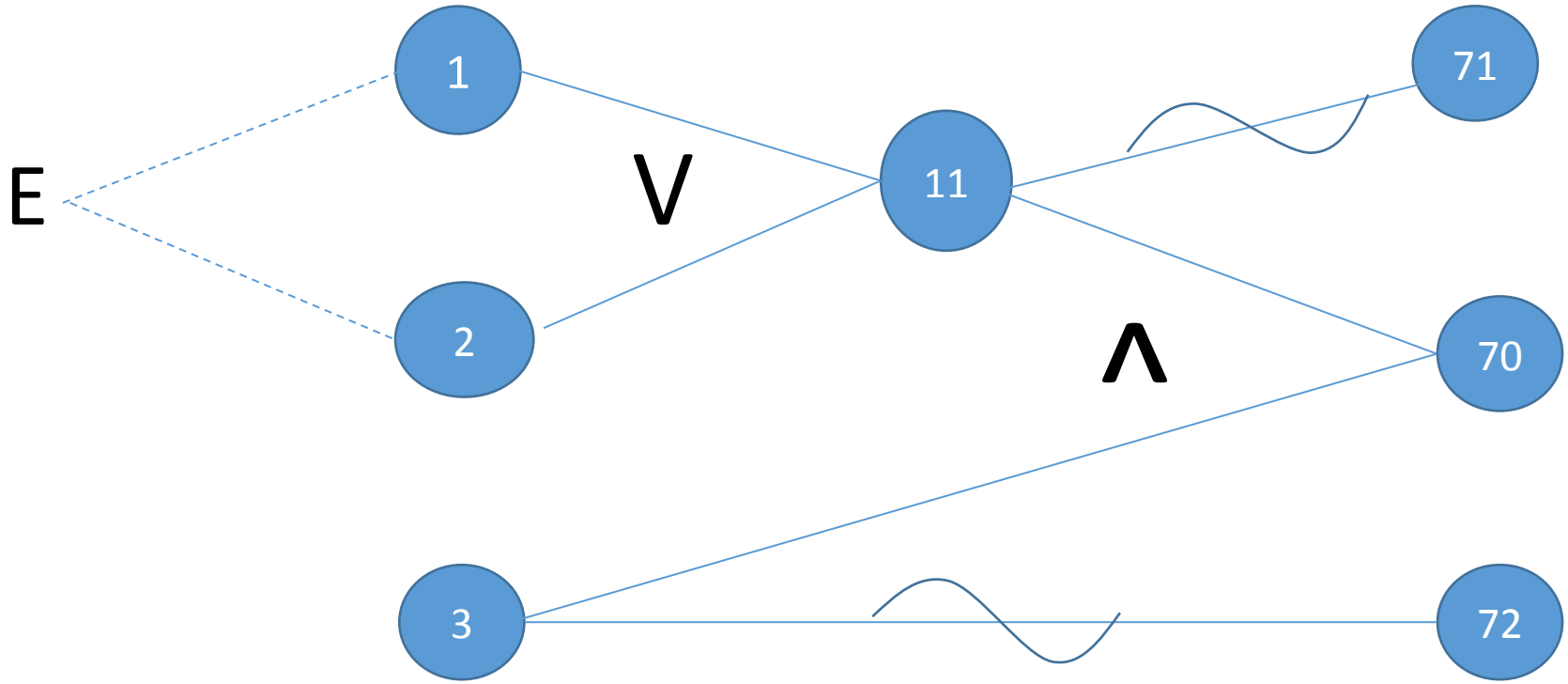


Таблица решений

Виды таблиц решений

- С ограниченными входами. Ответы на условия только Y, N. Выполняемые действия отмечаются X
- С расширенными входами. Ответом может быть любое значение. Действия можно именовать
- Полная таблица решений. Содержит все возможные комбинации условий 2 ^ N, где N – число условий

Пример таблицы решений

A	Начало	Y	N	N	N	N
B	Число обменов = 0	–	–	N	Y	–
C	Индекс массива в допустимых пределах	–	Y	N	N	Y
D	Элементы массива не упорядочены	–	Y	–	–	N
E	Установить начальный флажок	X	–	–	–	–
F	Число обменов = 0. Индекс массива = 1	X	–	X	–	–
G	Произвести обмен элементов. Установить флажок обмена	–	X	–	–	–
H	Увеличить индекс массива	–	X	–	–	X
I	Выход	–	–	–	X	–

Комментарий к таблице решений

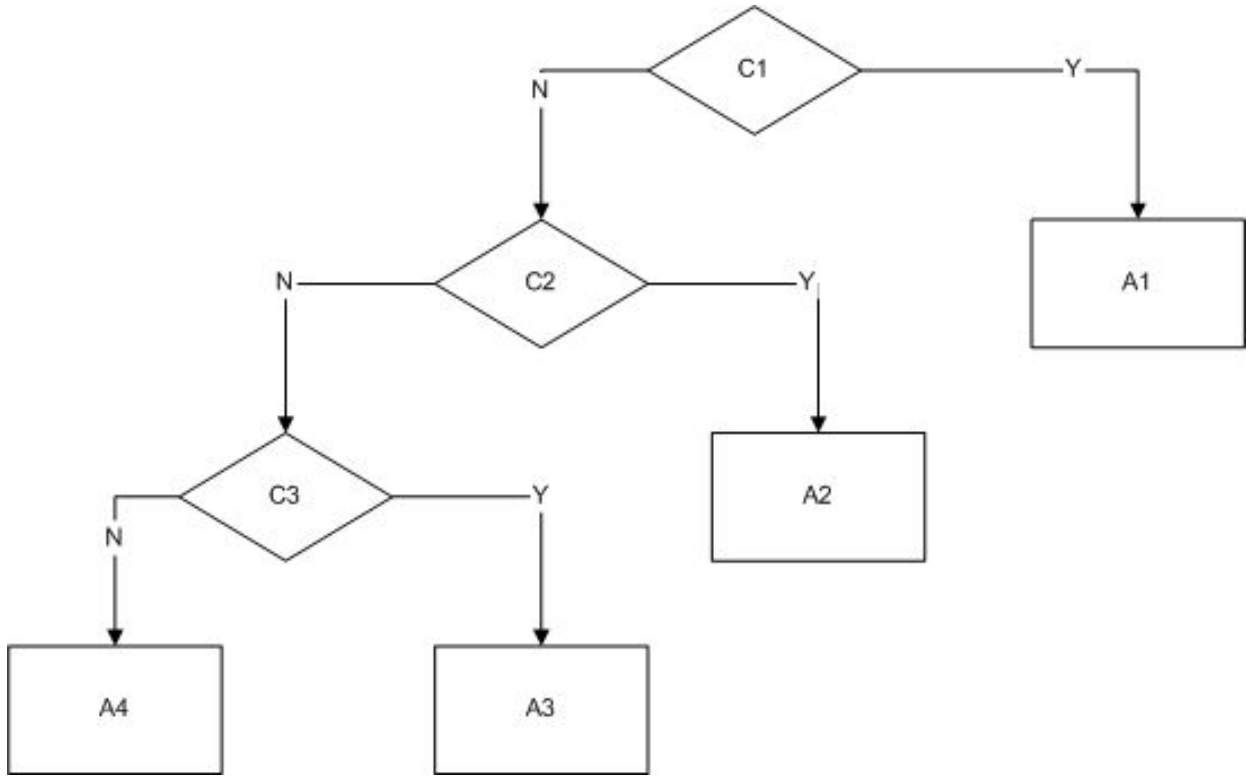
	Начало	+				
	Обработка элементов		+			+
	Окончание обработки и переход к повторной обработке			+		
	Окончание обработки и выход				+	

Реализация программы на основе таблицы решений

- Построение блок-схемы по таблице решений
- Выполнение программы на основе таблицы решений

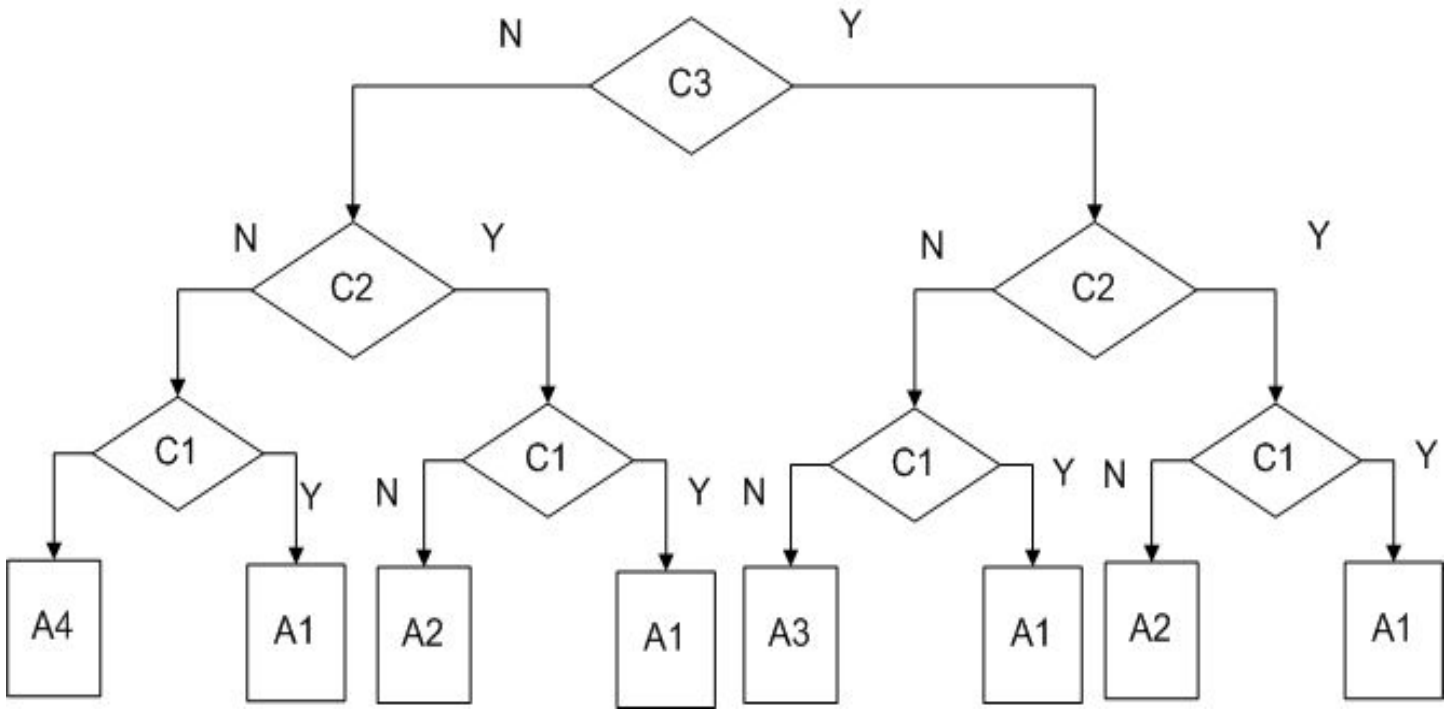
Построение блок-схемы по таблице решений

C1	Y	N	N	N
C2	-	Y	N	N
C3	-	-	Y	N
	A1	A2	A3	A4



Построение блок-схемы по таблице решений

C1	Y	N	N	N
C2	-	Y	N	N
C3	-	-	Y	N
	A1	A2	A3	A4

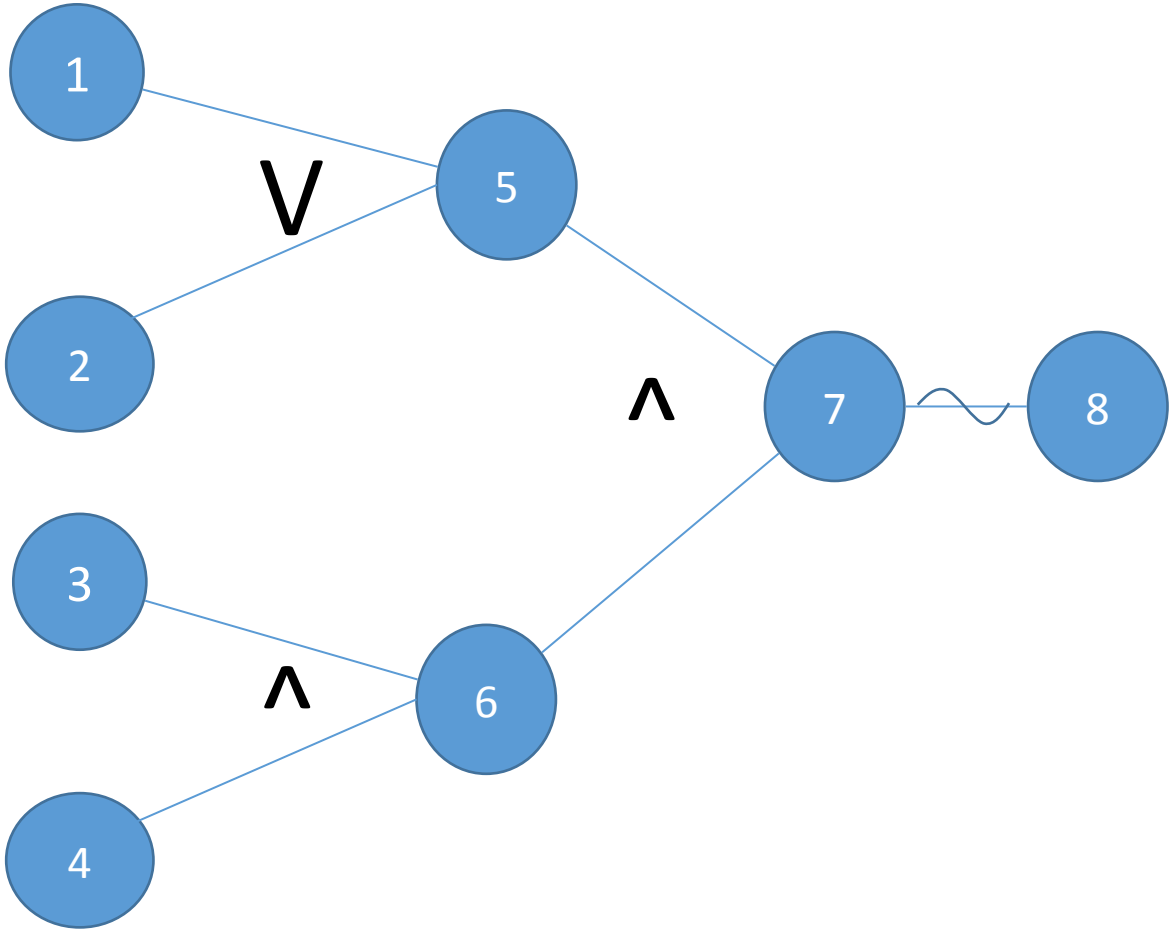


Преобразование причинно-следственной диаграммы в таблицу решений

- Причины – это условия в таблице. Следствия – это действия

1. Выберите следствие, которое должно находиться в состоянии 1
2. Продвигаясь в обратном направлении по диаграмме от следствия к причинам, найдите все комбинации причин, которые устанавливают данное следствие в 1
3. Создайте столбец в таблице решений для каждой комбинации причин
4. Определите для каждой комбинации причин состояния других следствий и поместите их в соответствующий столбец таблицы

Пример



1	1	1	1	0	0	0	1	1	1	0	0	0	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0
3	0	0	1	0	0	1	0	0	1	1	0	1	0
4	0	1	0	0	1	0	0	1	0	1	1	0	0
5	1	1	1	1	1	1	1	1	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1	1	1	1	1	1

Правила выбора комбинаций

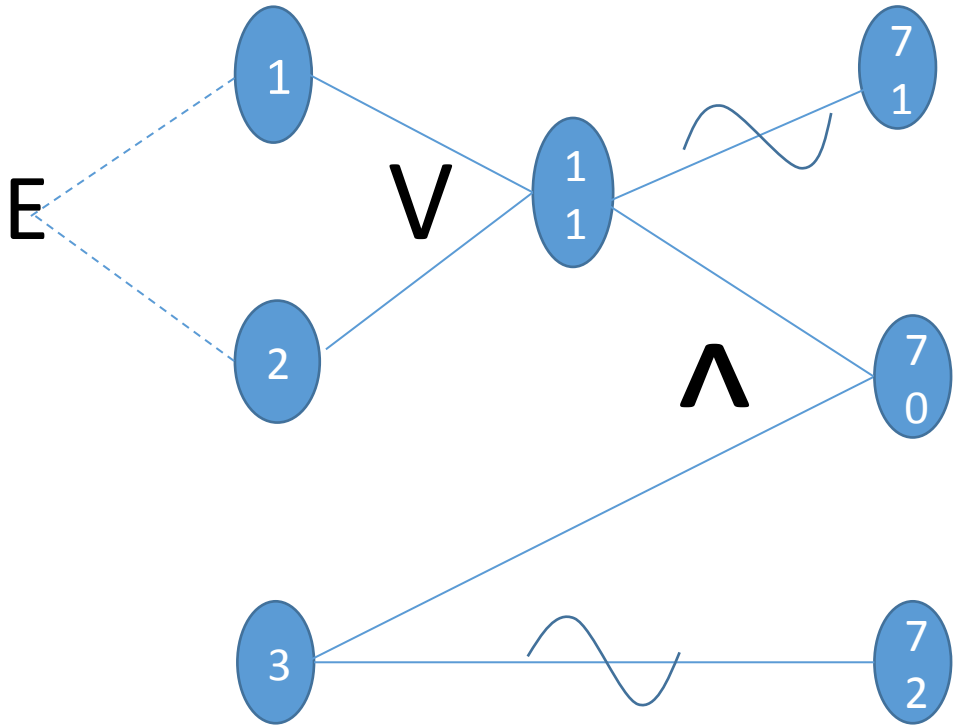
1. Если путь обратной трассировки проходит через узел OR, выход которого равен 1, то не следует устанавливать в 1 более одного входа
2. Если путь обратной трассировки проходит через узел AND, выход которого равен 0, то рассмотрите все комбинации входов, приводящих к 0. Если какие-то входы равны 1, то не обязательно перечислять все комбинации, приводящие к значениям 1 на этих входах
3. Если путь обратной трассировки проходит через узел AND, выход которого равен 0, то рассмотрите только одну комбинацию, при которой все входы равны 0

Применение правил выбора комбинаций

- Рассмотреть только один случай, когда узлы 5 и 6 равны 0 (правило 3). Можно выбрать любую из комбинаций 11,12,13
- Рассмотреть только один случай, когда узел 5 равен 1, если узел 6 равен 0 (правило 2). Вместо комбинаций 1,2,3 можно выбрать комбинации 4, 5, 6
- Комбинации 7, 8, 9 не удовлетворяют правилу 1
- Рассмотреть только один случай, когда узел 6 равен 1, если узел 5 равен 0 (правило 2)
- В таблице выбранные комбинации помечены «X»

1	1	1	1	0	0	0	1	1	1	0	0	0	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0
3	0	0	1	0	0	1	0	0	1	1	0	1	0
4	0	1	0	0	1	0	0	1	0	1	1	0	0
5	1	1	1	1	1	1	1	1	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1	1	1	1	1	1
											-	-	X
	X	X	X	-	-	-	-	-	-				
										X			

Пример разработки тестов



	1	2	3	4	5
1	0	0	1	0	1
2	0	1	0	1	0
3	1	1	1	0	0
11	0	1	1	1	1
71	1	0	0	0	0
70	0	1	1	0	0
72	0	0	0	1	1
Тесты	C1	B1	A2	BC	Тест не используется по правилу 2

Метод прогнозирования ошибок

- Некоторые люди обладают умением находить ошибки и без привлечения какой-либо методологии тестирования. Они подсознательно применяет метод проектирования тестов, называемый **прогнозирование ошибок**
- При наличии определенной программы опытный специалист интуитивно предполагает вероятные типы ошибок и затем разрабатывает тесты для их обнаружения
- Процедуру для данного метода описать трудно, так как метод в значительной степени является интуитивным
- Основная идея метода заключается в том, чтобы составить список возможных ошибок или ситуаций, в которых они могут появиться, а затем на основе этого списка написать тесты
- Например, такая ситуация возникает при значении 0 на входе и выходе программы. Следовательно, можно построить тесты, для которых определенные входные данные имеют нулевые значения и для которых определенные выходные данные устанавливаются в 0
- Если на вход программы подается переменное количество значений, то случаи отсутствия значений или одного значения с большой вероятностью могут содержать ошибки
- Если спецификации содержат некоторые моменты, не включенные в спецификацию в силу их очевидности для автора спецификации, то следует рассмотреть тесты, связанные с допущениями, которые программист может сделать при чтении спецификации

Пример теста на основе прогнозирования ошибок

- Для тестирования программы сортировки массива по возрастанию можно рассмотреть особые случаи, которые могли быть не учтены при разработке программы:
  - Сортируемый массив пуст
  - Массив состоит из одного элемента
  - Количество элементов массива четно (не четно)
  - Массив состоит из одинаковых элементов
  - Массив уже отсортирован
  - Массив отсортирован по убыванию