STAR Example Exam 2025

Exercise 1 (Risk management) 1. (2pt) Name a disadvantage of using "probability times impact" as a definition for risk.

Different projects and people weigh probability and impact differently. For example, in a nuclear power plant, low probability high impact risks are much more important to treat than high probability low impact risks. In a video game, this will be the other way around. When risk is defined as probability times impact, these two types of risks can no longer be distinguished.

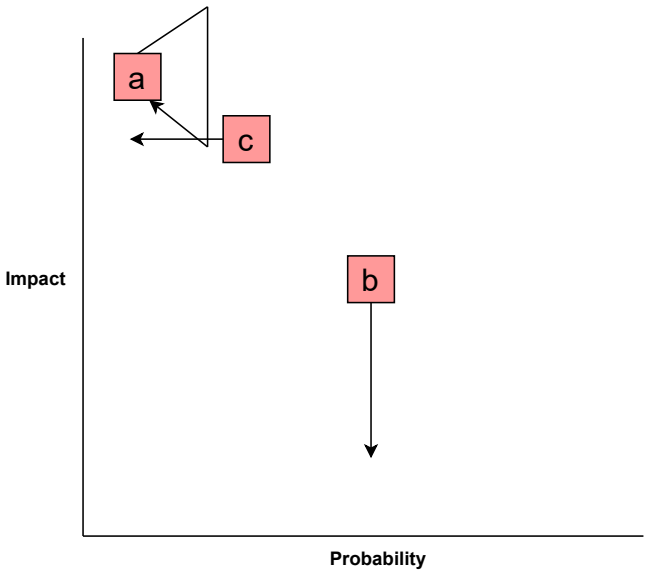
2. (6pt) Suppose your objective is "get a passing grade for the STAR exam". Give three risks associated with this objective, and for each risk a strategy to handle that risk. State under which of the main risk handling strategies your strategy falls, and sketch the risks and their strategies on the risk matrix.

Consider the following three risks:

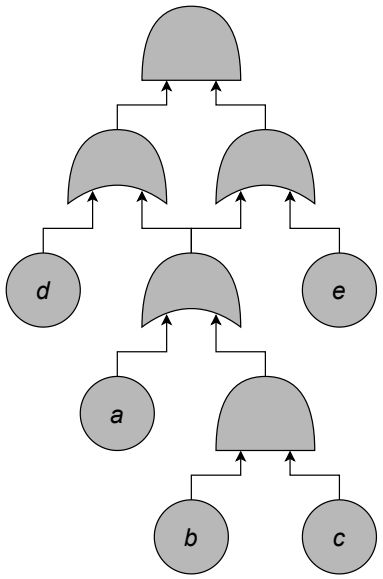
(a) My alarm does not go off, causing me to sleep in and be late for the exam (low probability, high impact). Since the exam is in the afternoon, I am willing to take this risk (tolerate).

(b) Questions about dynamic fault trees are difficult and I might not score well on them (medium probability, medium impact). I choose to study extra hard on the other topics, to make sure I can pass the exam even if I fail the dynamic fault tree questions (treat: reduce impact).

(c) I might be sick, so I cannot come to the exam (low probability, high impact). I make sure to sleep and eat well in the days before, to ensure I am in a good condition (treat: reduce probability).



Exercise 2 (Static fault trees) Consider the fault tree below.



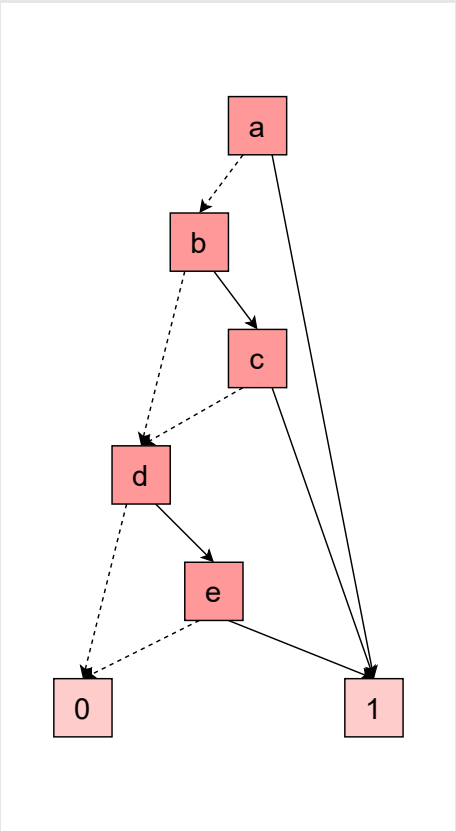
1. (2 point) List all of its minimal cut sets.

   The minimal cut sets are $\{a\}$, $\{b, c\}$, and $\{d, e\}$.

2. (2 point) Suppose $p_a = p_b = \frac{1}{2}$ and $p_c = p_d = p_e = \frac{1}{4}$. Approximate the failure probability using the cut set method; explain your answer.

   These cut sets have probability $\frac{1}{2}$, $\frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$, and $\frac{1}{4} \cdot \frac{1}{4} = \frac{1}{16}$, so the failure probability is approximately $\frac{1}{2} + \frac{1}{8} + \frac{1}{16} = \frac{11}{16}$.

3. (4 points) Give the binary decision diagram of this fault tree, with variable ordering $a < b < c < d < e$.



4. (3 points) Use the binary decision diagram to calculate the failure probability of the fault tree; explain your answer.

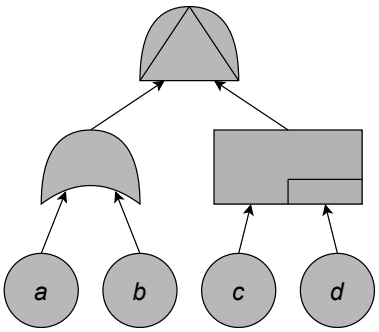
This BDD has the following paths from the root to 1:

Path	probability	
$a \to 1$	$\frac{1}{2}$	$\frac{1}{2}$
$a \to b \to c \to 1$	$\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{4}$	$\frac{1}{16}$
$a \to b \to d \to e \to 1$	$\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{4}$	$\frac{1}{64}$
$a \to b \to c \to d \to e \to 1$	$\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{4}$	$\frac{3}{256}$

So the failure probability is

$$\frac{1}{2} + \frac{1}{16} + \frac{1}{64} + \frac{3}{256} = \frac{128 + 64 + 16 + 3}{256} = \frac{211}{256} \approx 0.824.$$

Exercise 3 (Dynamic fault trees) Consider the dynamic fault tree below. The BEs a, b, c have failure rates $\lambda_a, \lambda_b, \lambda_c$, respectively, while d has failure rate λ_d until it takes over the function of c, when it will have failure rate λ_d.



1. (4 points) First consider just the OR-gate. Given a time $t \geq 0$, find an expression for the probability that the OR-gate has failed, in terms of λ_a, λ_b, and t. Explain your answer.

   For each gate g, let X_g be the random variable representing its failure time. If we let the OR-gate be v, then $X_v = \min(X_a, X_b)$. Hence

   $$\begin{aligned}\mathbb{P}(X_v \leq t) &= 1 - \mathbb{P}(X_v > t) \\ &= 1 - \mathbb{P}(X_a > t \text{ and } X_b > t) \\ &= 1 - \mathbb{P}(X_a > t)\mathbb{P}(X_b > t) \\ &= 1 - \mathrm{e}^{-\lambda_a t}\mathrm{e}^{-\lambda_b t} \\ &= 1 - \mathrm{e}^{-(\lambda_a+\lambda_b)t}.\end{aligned}$$

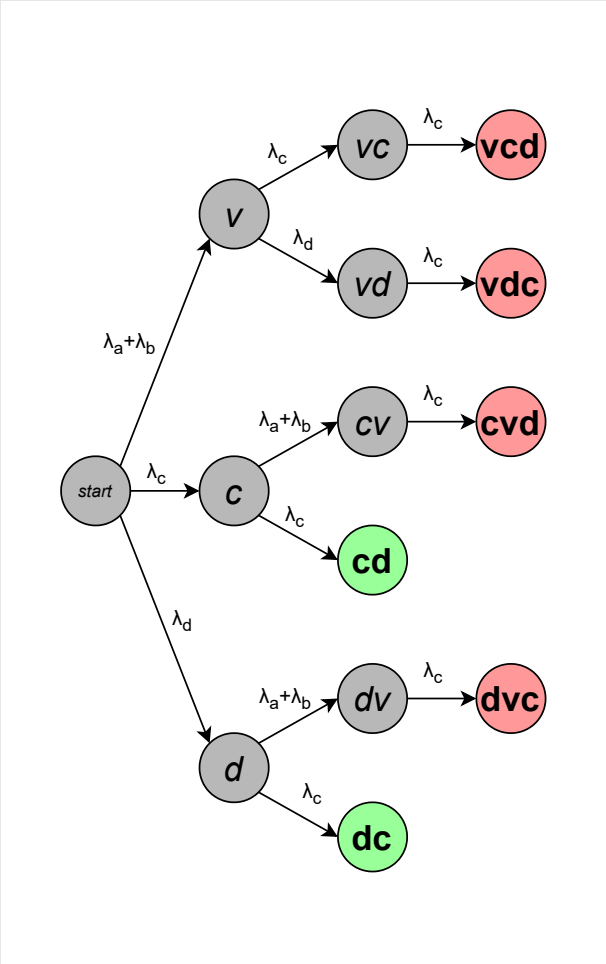
   Note that in the third step we used the fact that X_a and X_b are independent.

2. (1 point) Show that the failure time of the OR-gate is exponentially distributed. What is its failure rate?

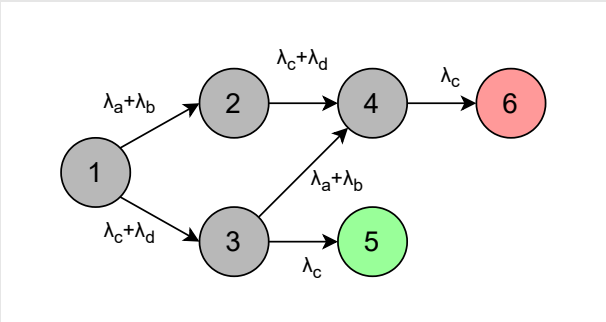
   This follows immediately from the previous exercise, which is an exponential distribution with failure rate $\lambda_a + \lambda_b$.

3. (6 points) Represent this DFT with a Markov chain with at most 6 states, and give its transition matrix.

First, observe that once a has failed, b is irrelevant, and the other way around. Therefore, we just focus on the failure of v, which has failure rate $\lambda_a + \lambda_b$ per the previous exercise. This leads to the following CTMC:



This can be reduced by merging states. We merge the failure states vcd, vdc, cvd, dvc to state 6 below, and the nonfailure states cd, dc to state 5 below. The states vc, vd, cv, dv now all have the same successor states with the same failure rate (i.e. they are equivalent), so we merge them to state 4 below. Finally, states c and d are now equivalent, so we merge them to state 3 below. This leads to the following Markov chain with 6 states:



Its transition matrix is

$$\begin{pmatrix} -\lambda_a-\lambda_b-\lambda_c-\lambda_d & \lambda_a+\lambda_b & \lambda_c+\lambda_d & 0 & 0 & 0 \\ 0 & -\lambda_c-\lambda_d & 0 & \lambda_c+\lambda_d & 0 & 0 \\ 0 & 0 & -\lambda_a-\lambda_b-\lambda_c & \lambda_a+\lambda_b & \lambda_c & 0 \\ 0 & 0 & 0 & -\lambda_c & 0 & \lambda_c \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Exercise 4 (Integration testing, old SET exam question) Suppose that you have a partial software implementation of the El Dorado game (for the rules of the game, see: `https://www.ultraboardgames.com/the-quest-for-el-dorado/game-rules.php`). You just finished your first implementation of the Hand cards of a Player, the Market Board, and that a Player can buy a card from the Market Board.
In Figure 1 some of the implementation's classes with the headers of some of its methods and constructors are shown in the Java language. Above each method a short description of its functionality is given in comments.

```
1  class MarketBooard {
2    // Constructor for initializing board with cards at start of game
3    MarketBoard()
4
5    // Iterate over all cards available for buying in the market
6    Iterator<Card> availableCardsIterator()
7
8    // Removes card from market
9    removeCard(Card card)
10 }
11
12 class HandCards {
13   // Constructor for initializing hand with random cards at start of game
14   HandCards()
15
16   // Constructor for initializing hand with t Traveler cards (each worth 1 gold)
17   // and random other cards at start of game
18   HandCards(int t)
19
20   // Removes gold cards and returns true if amount of gold is in hand
21   // returns false otherwise
22   boolean spendGold(int amount)
23 }
24
25 class Card {
26   //Creates a card with given name, power, color, and price
27   Card(String name, int power, Color color, int Price)
28
29   // Returns price of the card in gold
30   int getPrice();
31 }
32
33 class Player {
34   // Initializes player with hand
35   Player(HandCards hand)
36
37   // Returns the total amount of gold present in the hand
38   int goldInHand()
39
40   // Calls HandCards.spendGold MarketBoard.removeCard to buy card from market
41   // Returns true if hand has enough gold and false otherwise
42   boolean buy(Card card, MarketBoard market)
43 }
```

Figure 1: This figure shows some El Dorado classes with the headers of some of its methods and constructors. Above each method/constructor a short description of its functionality is given in comments.

Now you want to write an integration test for buying a card with hand cards from the market board. It should be checked that the gold spend from the hand of the Player is the same as the cost of the card from the market board.

(a) To implement the integration test, (3pts)

(i) which classes should to be replaced by stubs? Shortly motivate your answer.

(1pt) Only `Card`, because this integration test should integrate `MarketBoard` and `HandCards`.
Alternative answer: Card and MarketBoard should be a stub, because the integration test should check that the hand spends the right amount of gold, i.e. as specified by market, hand and player.

(ii) which classes should be replaced by drivers? Shortly motivate your answer.

(1pt) None, because `Player` can be used as-is, since it is already implemented and controls its `hand` via the `buy` method. Other classes cannot be drivers.

(iii) which classes should be used in the integration test as-is, i.e. with their actual implementation? Shortly motivate your answer.

(1pt) Player and Hand card should be used as-is, because it is about spending gold from the hand of a player.
Alternative answer: all classes, or all but card, see (i) for motivation

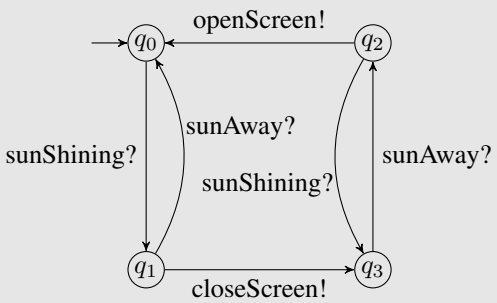
(b) Write Java code for the integration test, using methods/constructors of Figure 1, and any methods/-constructors for stubs or drivers that you need to introduce. If you include a line of code with a call to a stub or driver method/constructor, you need to include a comment line above that describes what that line of code achieves. (6pts)

```
1  //initializes market with a card costing 2 gold
2  MarketBoard market = new MarketBoard(new CardStub(2)); (1pt)
3  //adds 4 gold to  hand (1pt comments)
4  Player p = new Player(new HandCards(4)); (1pt)
5  int gold = p.goldInHand();
6  Iterator<> it = market.availableCardsIterator();
7  // gets card costing 2 gold
8  Card card = it.next(); (1pt)
9  assertTrue(p.buy(card, market)) (1pt)
10 assertEquals(card.getPrice,gold - p.goldInHand); (1pt)
```

Exercise 5 (LTS)

(a) Create an LTS that models an automatic sun screen that closes when sun is shining and opens when sun is not shining.

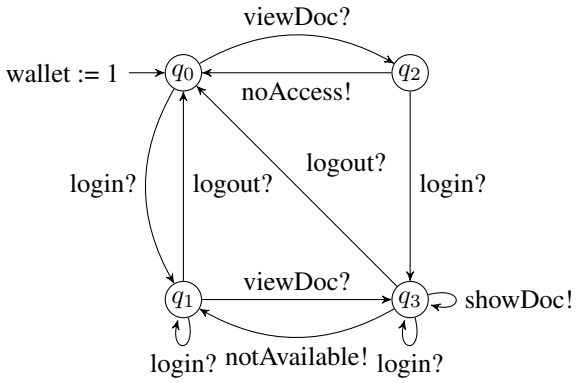
A possible solution:



(b) Which states of your LTS are quiescent?

States q_0 and q_3 (for the LTS given in the answer of a)

Exercise 6 (ioco) Consider the following LTS as a specification:

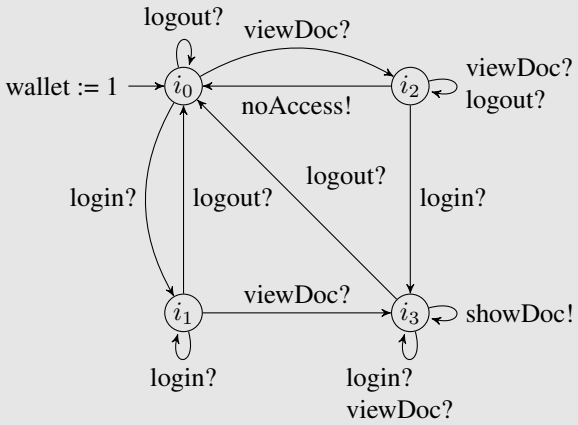


1. Create an implementation, that:

   - does not use one output label, that occurs on transition(s) of the specification, in any of its transition
   - is ioco with the specificaiton

   and explain why it is ioco conforming

A possible solution:



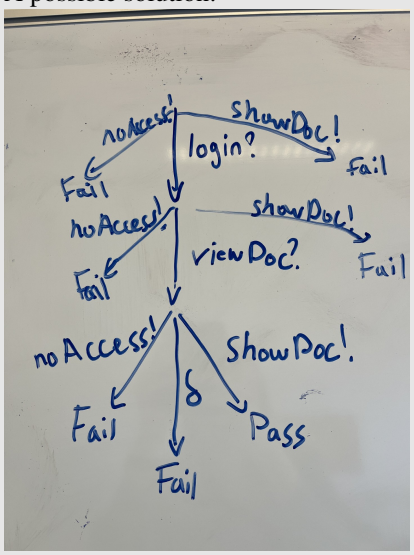
The implementation is the same as the given specification, except that:

- It has some additional input transitions to make the implementation input-enabled
- It does not have the notAvailable! output in i_3 while q_3 has this output

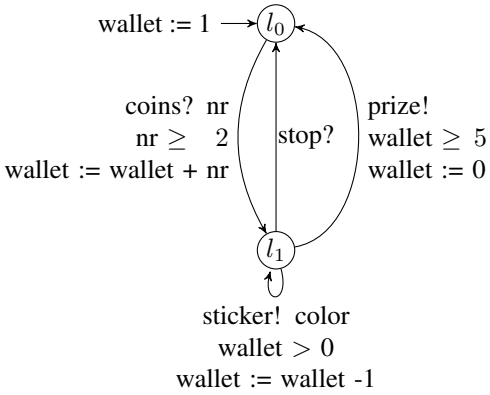
It is ioco conforming because $out(i_3) = \{\text{showDoc!}\} \subseteq \{\text{notAvailable!, showDoc!}\} = out(q_3)$. The extra input transitions do not make ioco false. The rest of the transitions are the same.

2. Create a test case of depth 3 (the tree has a trace with 3 subsequent labels), using batch test generation, such that the showDoc! output of the specification in state q_3 is covered by the test case.

A possible solution:



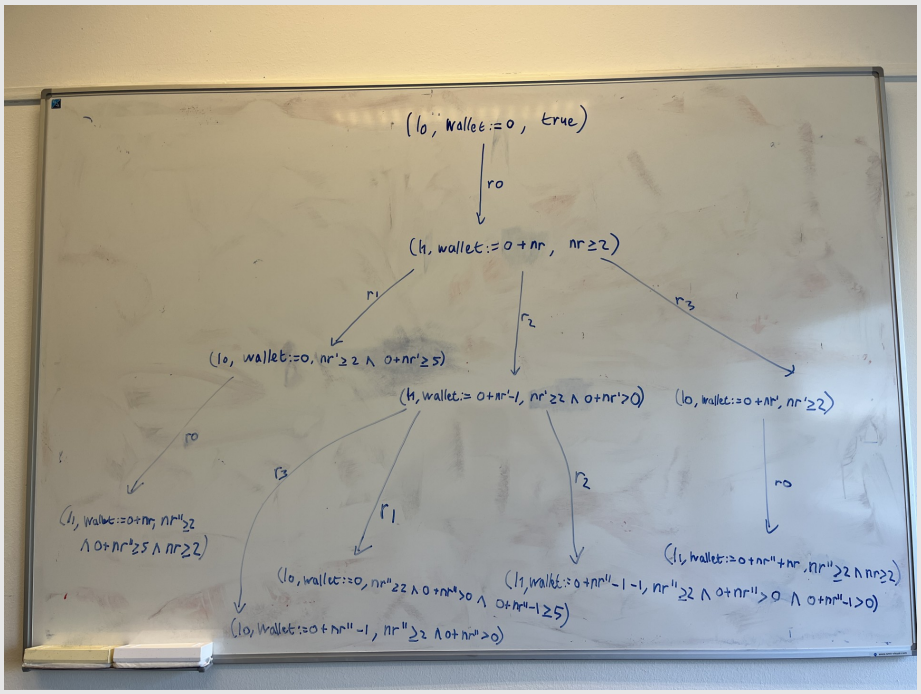
Exercise 7 (STS) Consider the following graphical representation of an STS in this exercise:



(a) Write down the formal definition of the switches of the STS

r_0 = (l_0, coins?, nr, nr $\geq$ 2, wallet := wallet + nr, l_1)
r_1 = (l_1, prize!, ϵ, wallet $\geq$ 5, wallet := 0, l_0)
r_2 = (l_1, sticker!, color, wallet $>$ 0, wallet := wallet -1, l_1)
r_3 = (l_1, stop?, ϵ, true, wallet := wallet, l_0)

(b) Write down the symbolic execution graph of the STS, up to level 3, i.e. take any valid sequence of three switches from the initial state of the graph.



(c) Write down a test trace such that the sticker switch is taken twice, and such that the prize! switch cannot be taken, according to the STS.

(coins? 4, sticker! ”blue”, sticker! ”red”)

(d) Point out to which branch of the symbolic execution graph your trace corresponds.

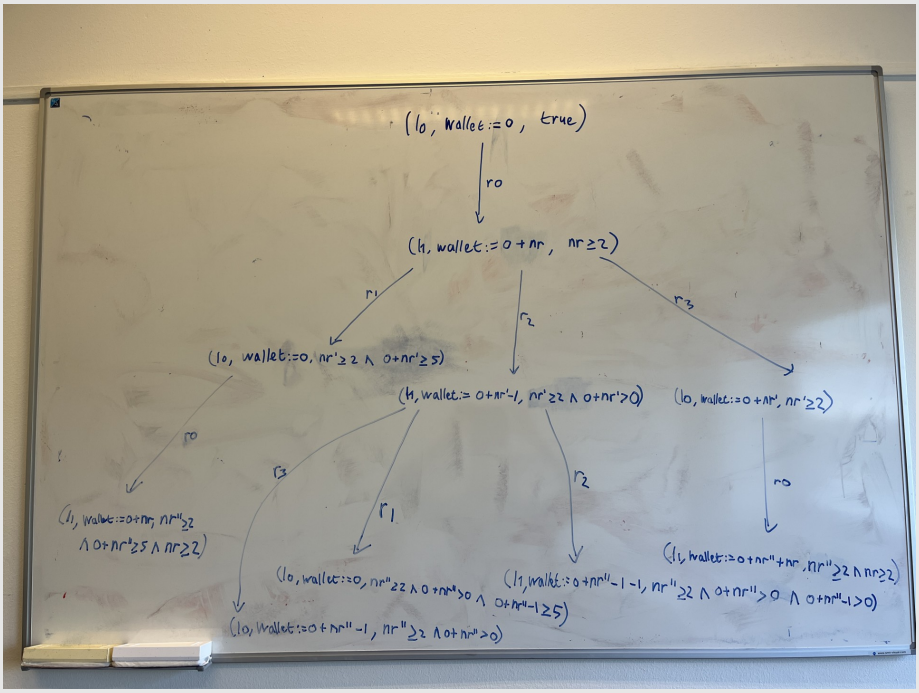
The branch taking switches $r_0 r_2 r_2$

(e) If you would choose values for test cases that test the first coins? switch, using boundary value analysis, what values would you then choose?

1 and 2
Alternative answer: 1, 2 and 3

(a) r_0 = (l_0, coins?, nr, nr ≥ 2, wallet := wallet + nr, l_1)
r_1 = (l_1, prize!, ϵ, wallet ≥ 5, wallet := 0, l_0)
r_2 = (l_1, sticker!, color, wallet > 0, wallet := wallet -1, l_1)
r_3 = (l_1, stop?, ϵ, true, wallet := wallet, l_0)

(b)



(c) (coins? 4, sticker! ”blue”, sticker! ”red”)

(d) The branch taking switches $r_0 r_2 r_2$