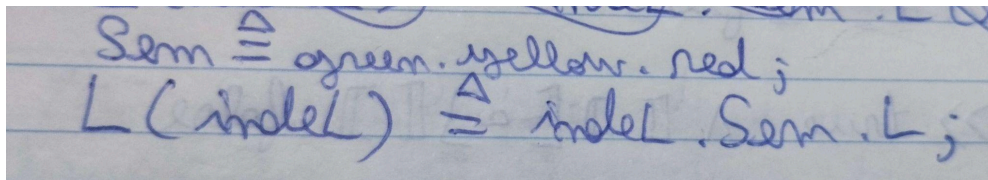


Realizado por:
Bruno Miguel Fernandes Araújo A97509

Problema 1:

(Eu não me recordava da preferência do educando, e acabei por deixar apenas o exc 1.1 e o 1.2 com explicações diretamente no documento.)

1.1)



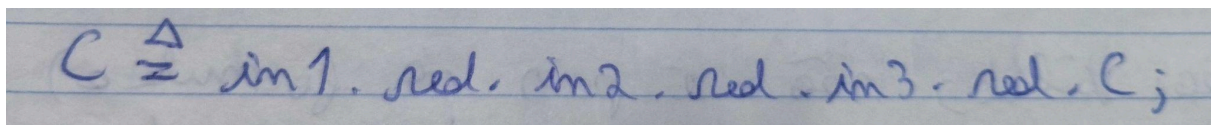
Handwritten code defining the `Sem` and `L` processes:

```
Sem  $\triangleq$  green. yellow. red;  
L(inlet)  $\triangleq$  inlet. Sem. L;
```

L tem um in que servirá como uma ação que permitirá distinguir as cópias deste entre si, ou seja A1, A2, A3 terão uma ação in própria.

Sem é um processo que tem apenas as ações das cores do semáforo e apenas existe para facilitar a interpretação.

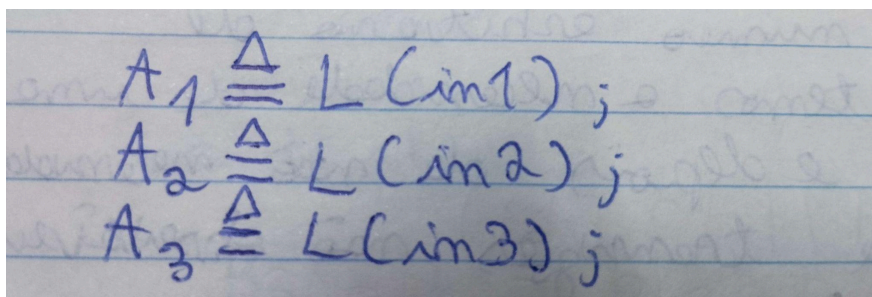
Por fim L tem uma chamada recursiva, pois este tem de ser um ciclo, "cycle".



Handwritten code defining the `C` process:

```
C  $\triangleq$  in1. red. in2. red. in3. red. C;
```

C tem as ações in associadas a cada A, acompanhadas com a red, que servirá como um out, ou seja depois desta terminamos o A em que se encontrava e começamos o próximo.



Handwritten code defining the `A1`, `A2`, and `A3` processes:

```
A1  $\triangleq$  L(in1);  
A2  $\triangleq$  L(in2);  
A3  $\triangleq$  L(in3);
```

Depois a chamada recursiva pois queremos que seja um "loop", que depois de A3 volte a repetir o processo começando novamente em A1.

1.2)

Código mcr12:

```
act green,yellow,red,in1,in2,in3,indeL;
```

```
proc
```

```
    Sem=green.yellow.red;
```

```
    L=indeL.Sem.L;
```

```
    A1=rename({indeL->in1},L);
```

```
    A2=rename({indeL->in2},L);
```

```
    A3=rename({indeL->in3},L);
```

```
    C=in1.red.in2.red.in3.red.C;
```

```
    T = allow({in1|in1,in2|in2,in3|in3,green,red|red,yellow},C||A1||A2||A3);
```

```
init T;
```

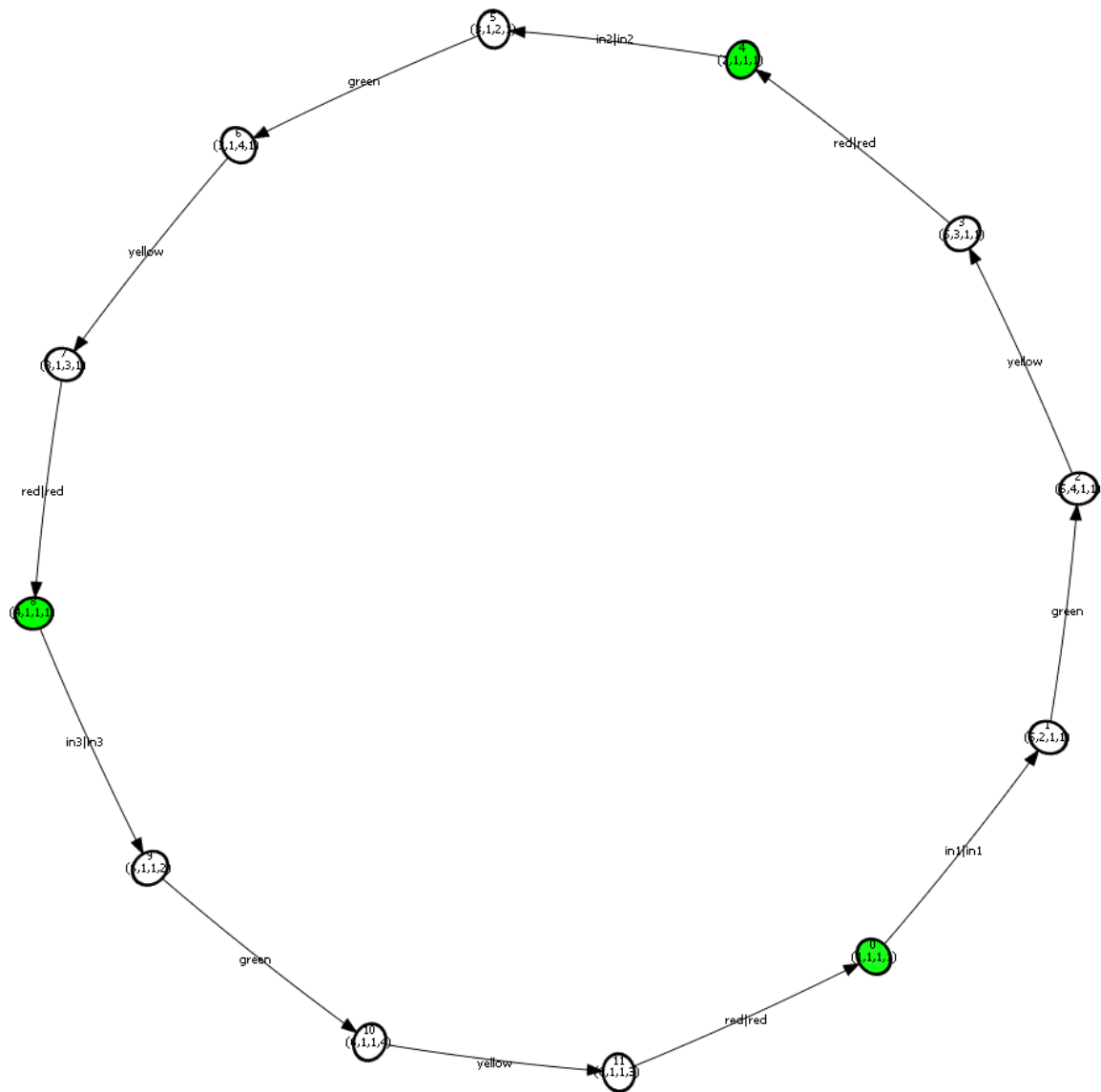
Sendo que em mcr12 não é possível um processo receber ações com parâmetros, demos uso á operação “rename” que nos permite alterar o indeL do processo L para in1 , in2 ou in3 dependendo do A em questão.

Para T temos de permitir as ações green , yellow e as sincronizações entre as ações in's e red de C com as dos A's.

Inicialmente tinha usado a ferramenta hide para tornar os in's não observáveis, mas como em mcr12 a ferramenta “allow” permite que apenas aquelas ações e sincronizações aconteçam, já não precisamos de ter cuidado com, por exemplo, a opção de ocorrer um in2 sozinho ou seja de acontecer o seguinte:

$(C | A_1 | A_2 | A_3)$
 $\downarrow \text{in2}$
 $(C | A_1 | \text{green, yellow, red, } A_2 | A_3)$

O sistema de transições obtido foi o seguinte:



Os estados com cor verde são aqueles que irão representar o início das transições de A1, A2,A3 em sincronização com C ou a green e yellow.

O mcr12 permite não só desenhar o sistema de transições como desenhá-lo com ou sem abstrações:

Sem abstrações temos:

Bissimulação forte.

Equivalência do traço.

Com abstrações temos:

Bissimulação ramificada.

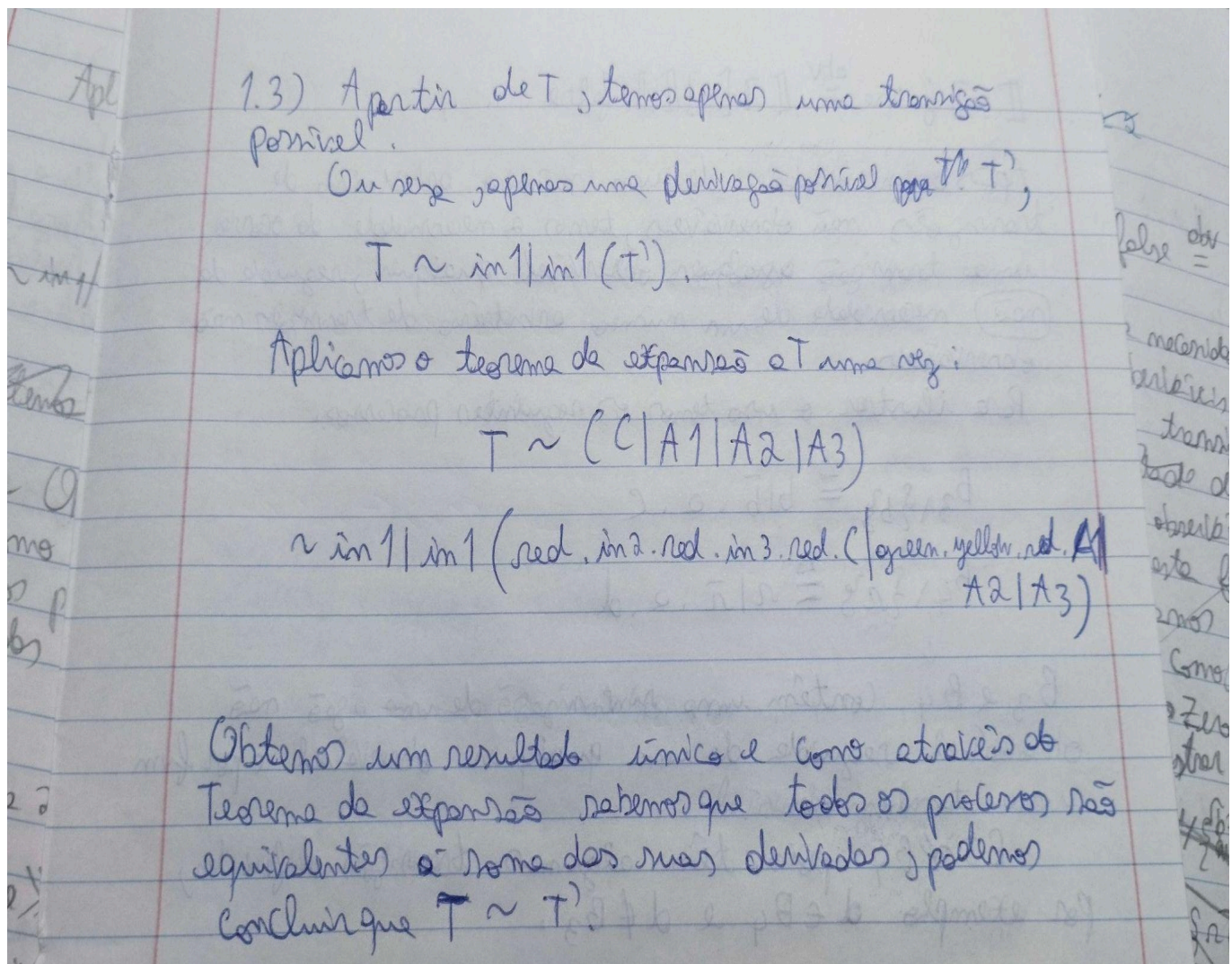
Divergência preservada da bissimulação ramificada.

Bissimulação fraca.

E por fim Equivalência do traço fraca.

Também permite simular a especificação, mas neste caso não traz muita utilidade pois cada estado tem apenas uma transição possível.

1.3)



Problema 2:

2.1)

$$2.1) \langle\langle fdee \rangle\rangle \text{ true} \stackrel{\text{adv}}{=} \langle\langle \rangle\rangle \langle fdee \rangle \langle\langle \rangle\rangle \text{ true}$$

Após a possibilidade de um número arbitrário de transições não observáveis, temos a possibilidade de transição observável $fdee$ seguida de, novamente, de possibilidade de um número arbitrário de transições não observáveis.

Para ilustrar o seu motor os seguintes processos:

$$B_1 \triangleq fdee$$

$$B_2 \triangleq fdee.b$$

Em ambos ocorre a transição $fdee$ e de $\langle\langle \rangle\rangle$ podemos ter 0 ou mais transições, então neste caso consideramos não ter ~~que~~ ter 0, além disso $B_1 \neq B_2$ pois a transição $b \notin B_1$ e como não têm as mesmas transições possíveis, não são bisimilares.

$$[-] \text{ false} \stackrel{\text{abv}}{=} [[-]] \text{ false}$$

Após a necessidade de um número arbitrário de transições não observáveis, temos a necessidade de observar uma transição qualquer observável qualquer, seguida de não necessidade de um número arbitrário de transições não observáveis.

Para ilustrar o uso temos os seguintes processos:

$$B_3 \setminus \{b_3\} \stackrel{\Delta}{=} b1\bar{b}.a.c$$

$$B_4 \setminus \{a\} \stackrel{\Delta}{=} a1\bar{a}.a.d$$

B_3 e B_4 contêm uma sincronização de uma ação não observável seguida de uma qualquer observável e por fim uma também observável.

$B_3 \neq B_4$ pois têm algumas transições diferentes, por exemplo $d \in B_4$ e $d \notin B_3$.

2.2)

$$2.2) \langle \langle \text{true} \rangle \rangle \wedge \llbracket -a \rrbracket \text{ false}$$

$$\langle \langle \rangle \rangle \wedge \langle \langle - \rangle \rangle \langle \langle \rangle \rangle \text{ true} \wedge \llbracket \rrbracket \llbracket -a \rrbracket \llbracket \rrbracket \text{ false}$$

Apesar de precisar valores, não é possível na fórmula a) após a ocorrência de um número arbitrário de transições não observáveis, temos a ocorrência de uma qualquer que não seja a e depois da não ocorrência de um número arbitrário de transições não observáveis.

$\langle \langle \rangle \rangle \langle \langle - \rangle \rangle \langle \langle \rangle \rangle$, não é suficiente para assegurar que a seja inevitável, nesta fórmula podemos ter por exemplo o seguinte problema:

$$E \triangleq E.a.O + E.c.O$$

Como podemos ver é possível evitar a, escolhendo a segunda opção do problema.

$$\llbracket \rrbracket \langle \langle - \rangle \rangle \text{ true} \wedge \llbracket -a \rrbracket \text{ false}$$

A fórmula b já corrige este problema com a presença de $\llbracket \rrbracket$ em $\llbracket \rrbracket \langle \langle - \rangle \rangle \text{ true}$.

Assim imediatamente após uma transição não observável é possível haver uma qualquer observável e com $\llbracket -a \rrbracket \text{ false}$ sabemos que tem de ser a.

Então a é inevitável

2.3)

2.3) É possível estabelecer uma relação entre a equivalência observacional e a equivalência ~~model~~ ^{mod}, pois como sabemos, dois protocolos não observacionalmente equivalentes se existir uma bisimulação fraca entre eles ($B_1 \approx B_2$), em que B_1 e B_2 são protocolos, uma bisimulação menos exigente que requer preservação apenas a preservação das transições dos estados, independentemente das ações envolvidas e essas transições.

Com esta nova lógica acrescentamos a capacidade de analisar transições não observacionais observáveis às fórmulas, abrindo um novo mundo de possibilidades de transição que não ~~eram~~ ^{são} possíveis para a bisimulação, ~~($\approx \subseteq \approx$)~~ mas sim para a bisimulação fraca.
($\sim \subseteq \approx$)