Industrial Processes Automation

MSc in Electrical and Computer Engineering
Scientific Area of Systems, Decision, and Control

Winter Semester 2018/2019

Group: C2
***70547** – João Ferreira*
***75268** – Rúben Tadeia*
***75987** – João Ribafeita*
***80978** – Gonçalo Pedro*

2nd Laboratory Assignment [1]

Handling Faults in Keyboard Reading

Part B - Properties of the PN

Q1: Draw the graph of the Petri net proposed in Part A, and save it as an XML file, using the Petri net editor available (PIPE2). Call the editor from Matlab with the command `pn_editor` use its file open user interface to work on the right file. Include in the report a PNG image, exported by PIPE2, showing the graph of the Petri net.

Resposta: Como visto na parte A deste laboratório, a abordagem escolhida para a nossa Petri Net consiste numa divisão em 3 grandes blocos.

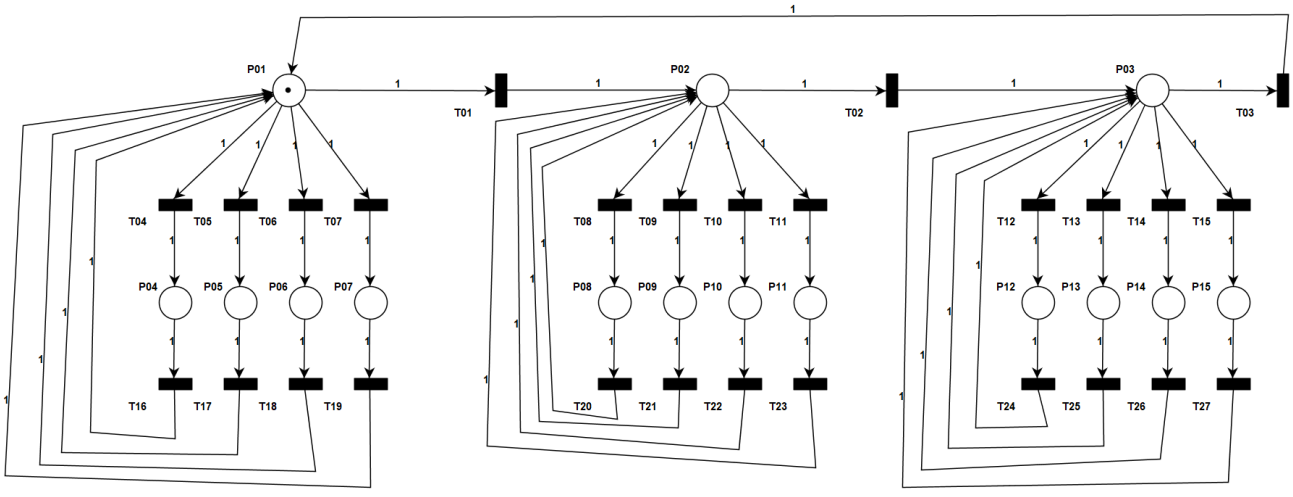


Figura 1 – Gráfico da Petri Net desenvolvida

Como se observa na *Figura 1*, cada bloco representa uma das 3 colunas do teclado. Em cada um dos estados espera-se que seja pressionada uma tecla na sua coluna respetiva. A partir deste momento o sistema evolui a partir do primeiro estado, em que espera por uma nova tecla premida. As transições

[1] Original guide by Prof. Paulo J. Oliveira. Revised by Prof. José Gaspar (2019).

T1, T2 e T3 correspondem a temporizadores usados neste DES. Estes temporizadores são a chave para a leitura correta de uma tecla em qualquer coluna. Adicionalmente e de modo a evitar ter uma petri net morta em que nenhuma transição seria disparada, foi adicionado um marker na primeira coluna.

Q2: Simulate a sequence of events such that there is a sequence of different states and the final state coincides with the initial one. Provide an interpretation of the sequence of events given the original assignment. The simulation can be done by inspection (is simple for a small number of places), using PIPE2 or using Matlab.

Resposta: Após ter corrigido alguns bugs do código da parte A, foi possível simular de novo a rede de Petri, com o conjunto predefinido de teclas. Neste caso, a simulação começa e termina sem existir nenhuma tecla pressionada, com a coluna 1 activada. Apresentam-se na figura seguinte os dados relevantes, nomeadamente as transições efectivamente disparadas, as teclas detectadas a cada momento e o estado do sistema.

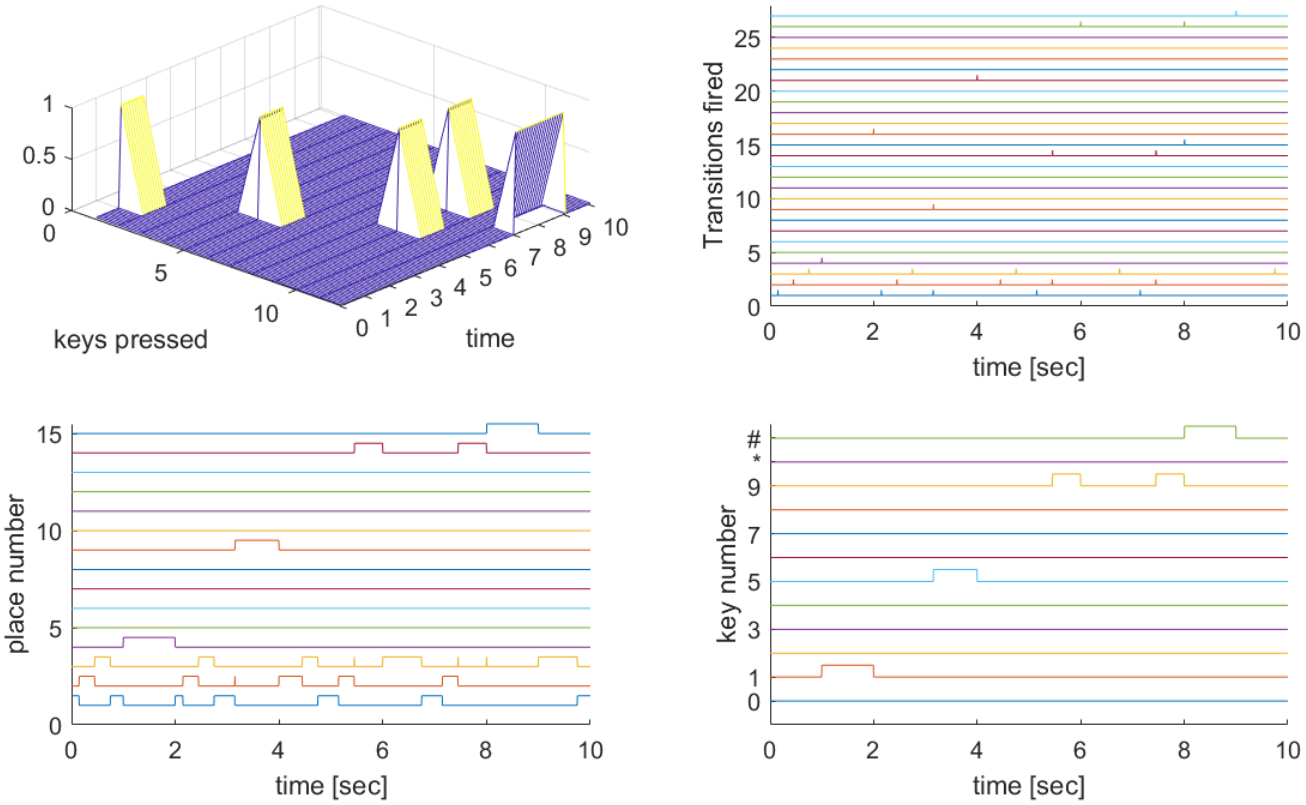


Figura 2 - Resultados da Simulação da Petri Net proposta

Na *Figura 2* observam-se os resultados obtidos na simulação da rede de Petri proposta para a leitura do teclado. A simulação foi alimentada pela seguinte lista de eventos (teclas pressionadas), em que x representa que nenhuma tecla foi pressionada e 12 representa a tecla #.:

$$x \rightarrow 1 \rightarrow x \rightarrow 5 \rightarrow x \rightarrow 9 \rightarrow x \rightarrow 9{,}12 \rightarrow x$$

É claro, no gráfico do canto inferior direito da *Figura 2*, que esta foi a sequência detetada na saída. O gráfico imediatamente ao lado suporta esta conclusão uma vez que as teclas que foram premidas foram: a tecla 1, a tecla 5, a tecla 9, e por fim as teclas 9 e #. Como nesta altura ainda não estava implementado, o algoritmo para resolução de teclas múltiplas, as primeiras teclas premidas seriam detetadas, o que resulta num pulso da tecla 9 seguido de um pulso da tecla #.

Q3: Discuss the properties that the proposed Petri net should verify and order the importance of those properties in the table below.

Order of importance	Property	Reason for Choice
1	Reachability	Representa todas as teclas que podem ser pressionadas no teclado
2	Liveness	Indica se é possível o uso contínuo do teclado ou se há limitação no número de disparos de alguma transição e/ou existência de deadlocks
3	Temporal Invariance	Indica se o funcionamento se mantém ao longo do tempo ou se se degrada
4	Boundedness	Indica o número máximo de marcas possíveis em cada lugar
5	Conservation	Indica se o número de marcas na rede de Petri se mantém constante
6	Safeness	Indica se é possível validar múltiplas teclas simultaneamente
7	Coverability	Indica se partindo de um estado arbitrário é possível chegar a qualquer outro estado

Resposta: A escolha para a ordem de importância das propriedades da petri net, assentou-se na prioridade das informações que queremos extrair de uma petri net assim que a queremos analisar. A razão da nossa escolha, assim como um breve enquadramento da relação que existe entre as propriedades e o seu funcionamento encontra-se na coluna adicional (terceira) da tabela anterior.

Q4: Study property #1, resorting to the methods studied in the course.

Resposta: A primeira propriedade escolhida foi a **Reachability** da Petri net.

Para melhor compreensão do estudo desta propriedade e das seguintes, vamos ter por base a nossa Petri net C = (P, T, I, O, μ_0). Em que P são os places/estados, T as transições e μ_0 é o marking inicial, que é $\mu_0 = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$. Tendo 15 estados e 27 transições, a reachability tree obtida pode ser observada na *Figura 3* e contém todos os estados atingíveis.

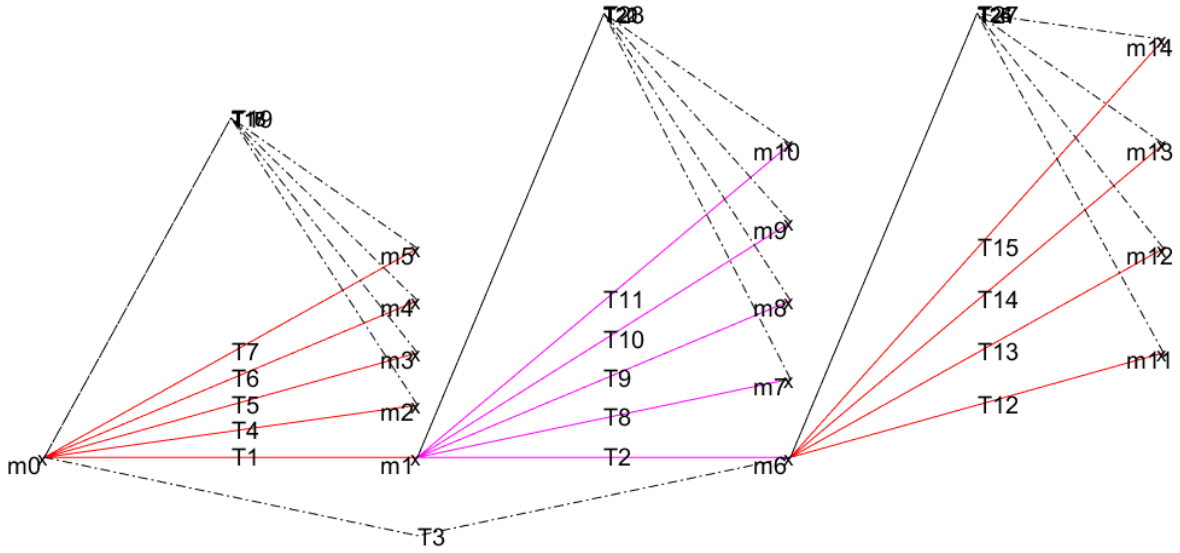


Figura 3 – Reachability Tree

Utilizando o software pn_editor conseguimos obter uma reachability tree um pouco mais esclarecedora e que pode ser consultada na *Figura 4*.

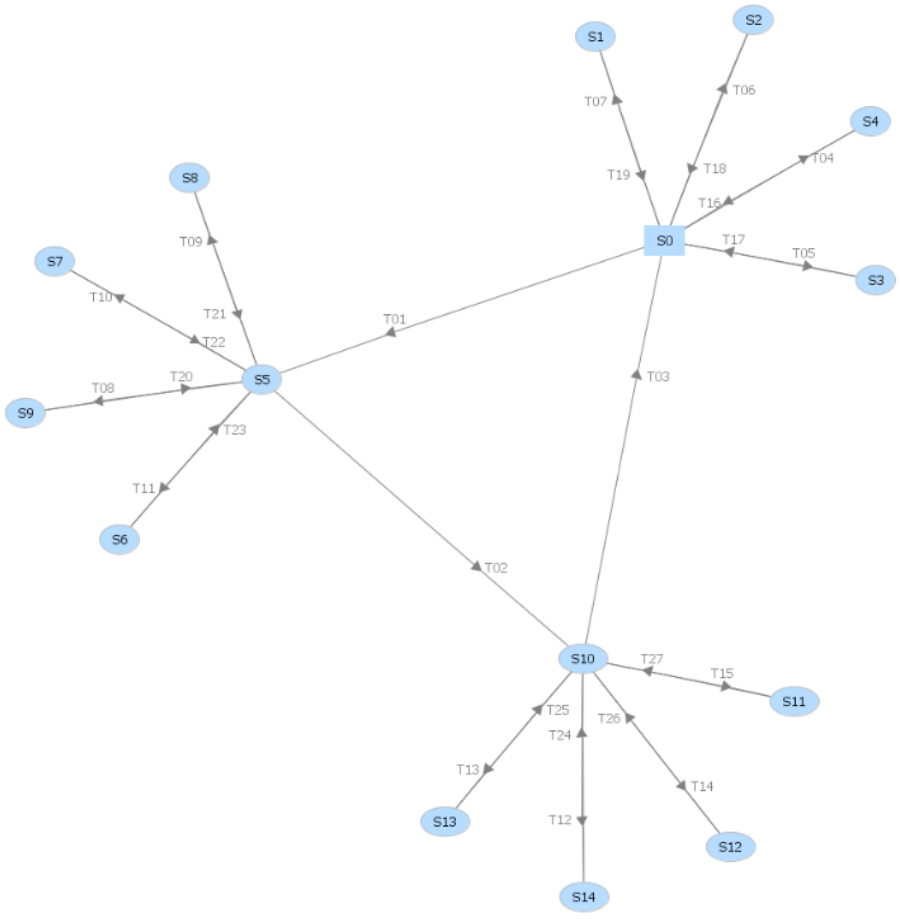


Figura 4 – Reachability Tree obtida automaticamente pelo PN_Editor (PIPE)

Sendo R(C, μ_0) a reachability na nossa Petri Net C com a marcação inicial μ_0 e sabendo que a evolução de estado da Petri Net é dada por:

$$\mu(k+1) = \mu(k) + Dq(k)$$

podemos concluir a partir da Reachability Tree e do marking inicial μ_0 que a rede C tem um conjunto finito de estados atingíveis e que é possível atingir todos os outros estados.
No nosso caso, a escolha da Petri net levou a que a Reachability Tree seja igual à Coverability Tree, isto porque nunca temos mais do que um marking por estado.

Q5: Study property #2, resorting to the methods studied in the course.

Resposta: A segunda propriedade escolhida foi a **Liveness** da Petri net. A análise da Liveness podia ser feita de um modo iterativo, ou seja, testando todas as transições e comparando se cumprem os requisitos de cada nível de "vivacidade". Quando se pretende estudar a Liveness, procura-se saber se uma dada transição pode voltar a ser disparada, em oposição ao estudar a Safety de uma petri net procura-se saber se dada transição não volta a ser disparada. Para determinação desse nível, analisou-se a reachability tree.

Analisando a reachability tree observa-se que, para cada marca é sempre possível atingir qualquer outra definindo uma sequência de disparos conveniente, concluindo-se que todas as transições podem ser disparadas. Adicionalmente verifica-se que dado um estado arbitrário é possível atingir qualquer outro, isto é, não existem deadlocks a prender a rede num estado. Concluíu-se portanto que a rede tem uma Liveness de nível 4.

Q6: Study property #3, resorting to the methods studied in the course.

Resposta: A terceira propriedade escolhida foi a **Temporal Invariance** da Petri net.

Para determinar os vectores responsáveis por tornar possível chegar de um a qualquer outro estado, resolvemos a equação seguinte.

$$Dq = 0$$

A matriz D é uma matriz com dimensões 15x27 e encontra-se nos anexos. (clique aqui para viajar até aos anexos). O vetor q é um vetor coluna com o número de linhas igual ao número de transições.

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{27} \end{bmatrix}$$

$$\begin{cases} (q_3 + q_{16} + q_{17} + q_{18} + q_{19}) - (q_1 + q_4 + q_5 + q_6 + q_7) = 0 \\ (q_1 + q_{20} + q_{21} + q_{22} + q_{23}) - (q_2 + q_8 + q_9 + q_{10} + q_{11}) = 0 \\ (q_2 + q_{24} + q_{25} + q_{26} + q_{27}) - (q_3 + q_{12} + q_{13} + q_{14} + q_{15}) = 0 \\ q_4 - q_{16} = 0 \\ q_5 - q_{17} = 0 \\ q_6 - q_{18} = 0 \\ q_7 - q_{19} = 0 \\ q_8 - q_{20} = 0 \\ q_9 - q_{21} = 0 \\ q_{10} - q_{22} = 0 \\ q_{11} - q_{23} = 0 \\ q_{12} - q_{24} = 0 \\ q_{13} - q_{25} = 0 \\ q_{14} - q_{26} = 0 \\ q_{15} - q_{27} = 0 \end{cases}$$

Resolver o sistema anterior permite obter um vetor q, que é uma condição necessária, no entanto não suficiente para provar que existe Temporal Invariance. Como temos mais incógnitas do que equações, utilizou-se um software de cálculo para obtenção de um vetor q, sendo um desses o que se segue.

$$\boldsymbol{q} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]^T$$

Era também fácil de notar que das três primeiras equações do sistema vinha:

$$\begin{cases} q_1 = q_3 \\ q_1 = q_2 \\ q_2 = q_3 \end{cases}$$

Podemos então concluir que a nossa Petri net é invariável no tempo. De notar que esta seria a resposta intuitiva, já que no modelo considerado se assume que a sequência de ações é: varrimento sequencial temporizado das colunas, sendo que a temporização cria um compasso de espera para uma tecla ser pressionada e após a tecla ser pressionada o estado volta ao varrimento começando na mesma coluna.

Q7: Study property #4, resorting to the methods studied in the course.

Resposta: A quarta propriedade escolhida foi a **Boundedness** da Petri net. Dada a nossa Petri net C, ela será K-bounded se $\mu_i' \leq k$, para todos os nossos:

$$\mu' = (\mu_1', \dots, \mu_{15}') \in R(C, \mu_0)$$

Assim sendo, e por inspeção da Petri net, reparamos que é K-bounded, porque todos os places são K-bounded. E é fácil obter o valor de K, dado que para qualquer estado nunca temos mais do que 1 marca,

vem que $K = 1$. Com este resultado, a Petri net é também Safe e Conservativa, como se verá mais adiante neste relatório.

Q8: Study property #5, resorting to the methods studied in the course.

Resposta: A quinta propriedade escolhida foi a **Conservation** da Petri net. A nossa Petri net C, será strictly conservative se para todos os valores de $\mu' = (\mu_1', \dots, \mu_{15}') \in R(C, \mu_0)$ se verificar a igualdade:

$$\sum_{p_i \in P} \mu'(p_i) = \sum_{p_i \in P} \mu(p_i)$$

No nosso caso, a Petri net é strictly conservative, tendo em conta que todas as transições têm no máximo 1 place/estado como input e 1 place/estado como output, não obstante de ser apenas "consumido" um marker no processo.

Q9: Study property #6, resorting to the methods studied in the course.

Resposta: A sexta propriedade escolhida foi a **Safeness** da Petri net. Dada a nossa Petri net C, ela será Safe se $\mu_i' \leq 1$, para todos os nossos:

$$\mu' = (\mu_1', \dots, \mu_{15}') \in R(C, \mu_0)$$

Tendo por base que a Petri net é K-bounded ou seja $\mu_i' \leq k$, só falta provar que $K = 1$. Para obter o valor de K, basta pensar que em qualquer estado nunca temos mais do que 1 marca. Com este resultado podemos concluir que todos os estados são safe. Como todos os places/estados são safe, a petri net também o é. Por exemplo, como não existem contadores na petri net não é possível aumentar infinitamente o número de markers de um place, o que a torna segura.

Q10: Study property #7, resorting to the methods studied in the course.

Resposta: A sétima propriedade escolhida foi a **Coverability** da Petri net. Dada a nossa Petri net C e o marking inicial μ_0, então podemos afirmar que se μ cobre μ' se para todos os places $p_i \in P$ temos:

$$\mu'(i) \leq \mu(i)$$

Na tentativa de responder a esta questão foi desenhado o coverability graph, não sendo explicitamente necessária, pois nenhum estado na nossa Petri net cobre outro estado qualquer.
Quase todas as Petri Nets que são seguras (Safe) (como a nossa), acabam por não ter coverability de estados.
O Coverability graph encontra-se nos anexos. (clique aqui para viajar até aos anexos).

Q11: *[Set the priority of transitions]* Many Petri net simulators do not randomize the firing of conflicting transitions. In the Petri net simulator used in the laboratory, conflicting transitions are prioritized by their identification numbers: the transition with the lowest number is the one selected to be fired.

(i) Let D^- and D^+ denote the preconditions and post-conditions matrices, and let q denote a firing vector. Describe how to obtain a new Petri net characterized by $D_2^- = f_1(D^-)$ and $D_2^+ = f_2(D^+)$, with a firing vector $q_2 = f_3(q)$ such that the original effects of transitions are kept but q_2 has the desired priorities. What are the functions $f_1(.)$, $f_2(.)$ and $f_3(.)$?

Suggestion: Consider that the functions $f_1(.)$, $f_2(.)$ and $f_3(.)$ can be implemented as matrices multiplying the input $(.)$ by its left or right. More precisely, define the functions by considering one multiplying matrix $P_{n,I}$ which operates permutations as defined in the next Matlab functioçn

```
function P= vector_permute( n, idx )
% usage example: q= (1:5)'; P= vector_permute( [1 4 2], 5 ); P*q
P= eye(n);
P= P( [idx setdiff(1:n, idx)], : );
```

Additional notes about the matrix $P_{n,I}$: n is the length of vector q, i.e. the number of columns of the D matrices; I is a list of indexes of q (transitions) to have priority, represents `idx` in the Matlab function; since $P_{n,I}$ consists just of line permutations of an identity matrix, it is always invertible.

Resposta: De modo a puder implementar as prioridades nos disparos, teve de ser criado código para implementar as funções $f_1(.)$, $f_2(.)$ e $f_3(.)$. A função f_3 implementa a permutação dos elementos do vector q_k. Esta função retorna uma matriz identidade (com colunas permutadas de acordo com os idx fornecidos) que, multiplicada à direita por q_k, efectua as permutações dos elementos prioritários (indicados por idx) para o início do vector.
As funções f_1 e f_2 implementam as permutações equivalentes na matriz de incidência (dividida em D^- e D^+), permutando as colunas correspondentes aos disparos prioritários (indicados por idx) para o início das matrizes.

(ii) Make a small Matlab demonstration illustrating the changing of the matrices and firing vector, such that one of the transitions is given priority in case of conflict. Suggestion: add to the function `PN_sim.m` a transition reordering input option and add code implementing $f_1(.)$, $f_2(.)$ and $f_3(.)$, such that the internal function `PN_state_step()` does not need to be changed. See more details in Appendix A.

Resposta: Implementando as funções anteriores e efectuando algumas alterações no código, é possível simular o sistema como pretendido. De modo a demonstrar o efeito da existência de prioridades, foi usado o mesmo conjunto de teclas que se tinha usado anteriormente e definida a transição 15 (#) como prioritária relativamente a todas as outras transições e não apenas às transições associadas à 3ª coluna. Analisando os resultados, verifica-se que, no caso do conflito, passa a ser disparada apenas a transição correspondente ao #, fazendo com que apenas o # seja detetado.

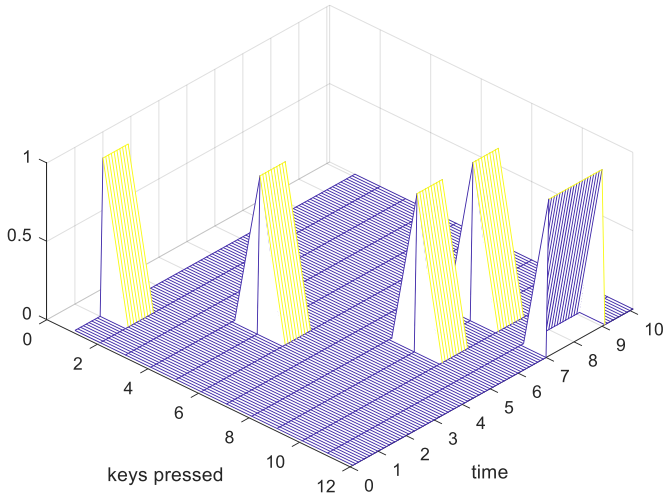


Figura 5 – Teclas premidas

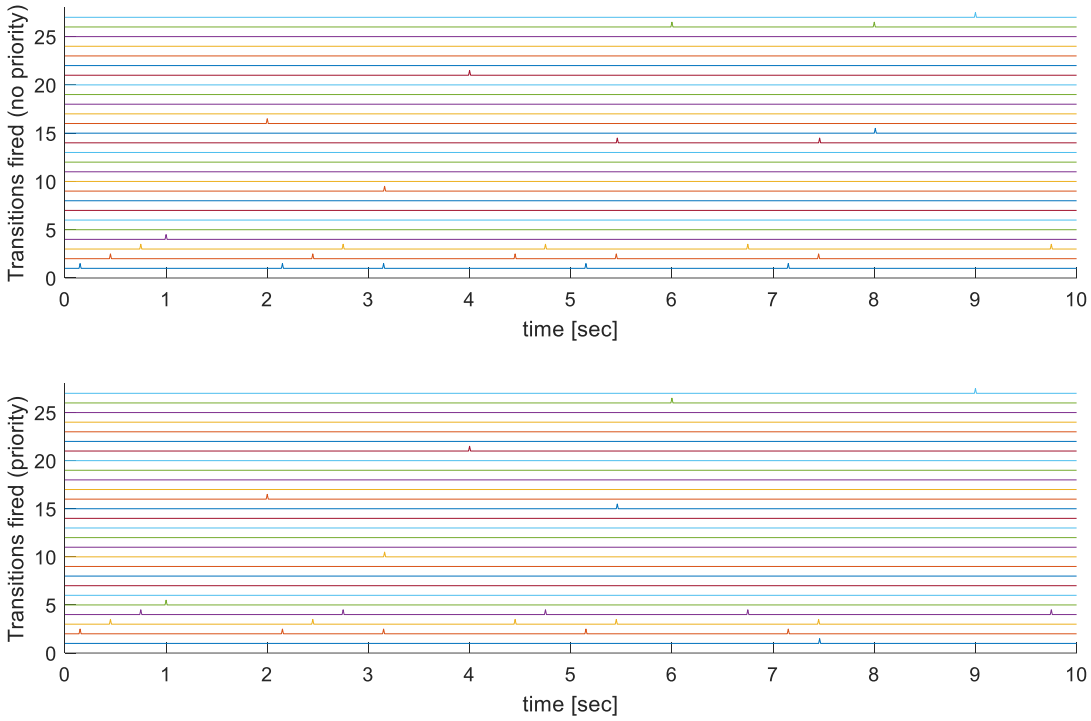


Figura 6 – Transições disparadas (para os casos com e sem prioridade definida)

Por fim, temos as *Figura 6* e *Figura 7* que mostram a evolução dos markings pelos estados, tanto para o caso sem prioridades, como para o caso da transição 15 (tecla #) ser prioritária. Relativamente à figura 7, na imagem da esquerda (sem prioridades) observamos que ao fim de 7 segundos foram premidas as teclas 9 e #, aparecendo o 9 e depois o # (depois de largar o 9, em *t=8s*) como detetados. Com a implementação do sistema de prioridades (imagem da direita), vemos que ao fim de 7 segundos se as teclas 9 e # forem premidas em simultâneo, apenas é detetada a tecla #.

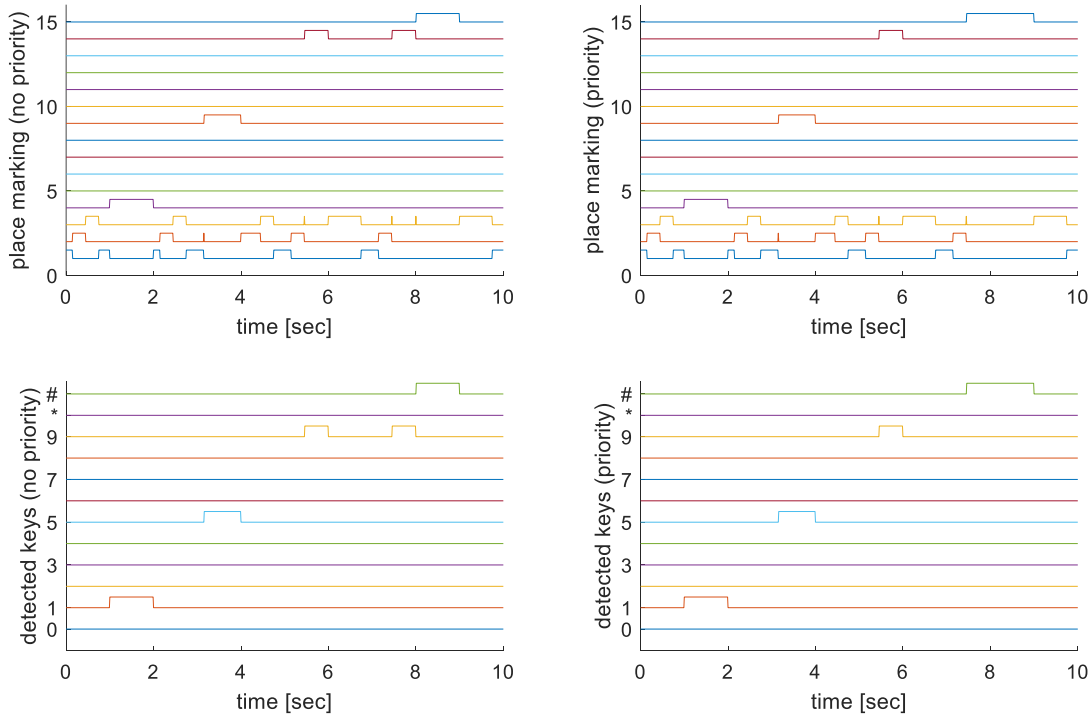


Figura 7 – Estado do sistema e teclas detectadas (para os casos com e sem prioridade definida)

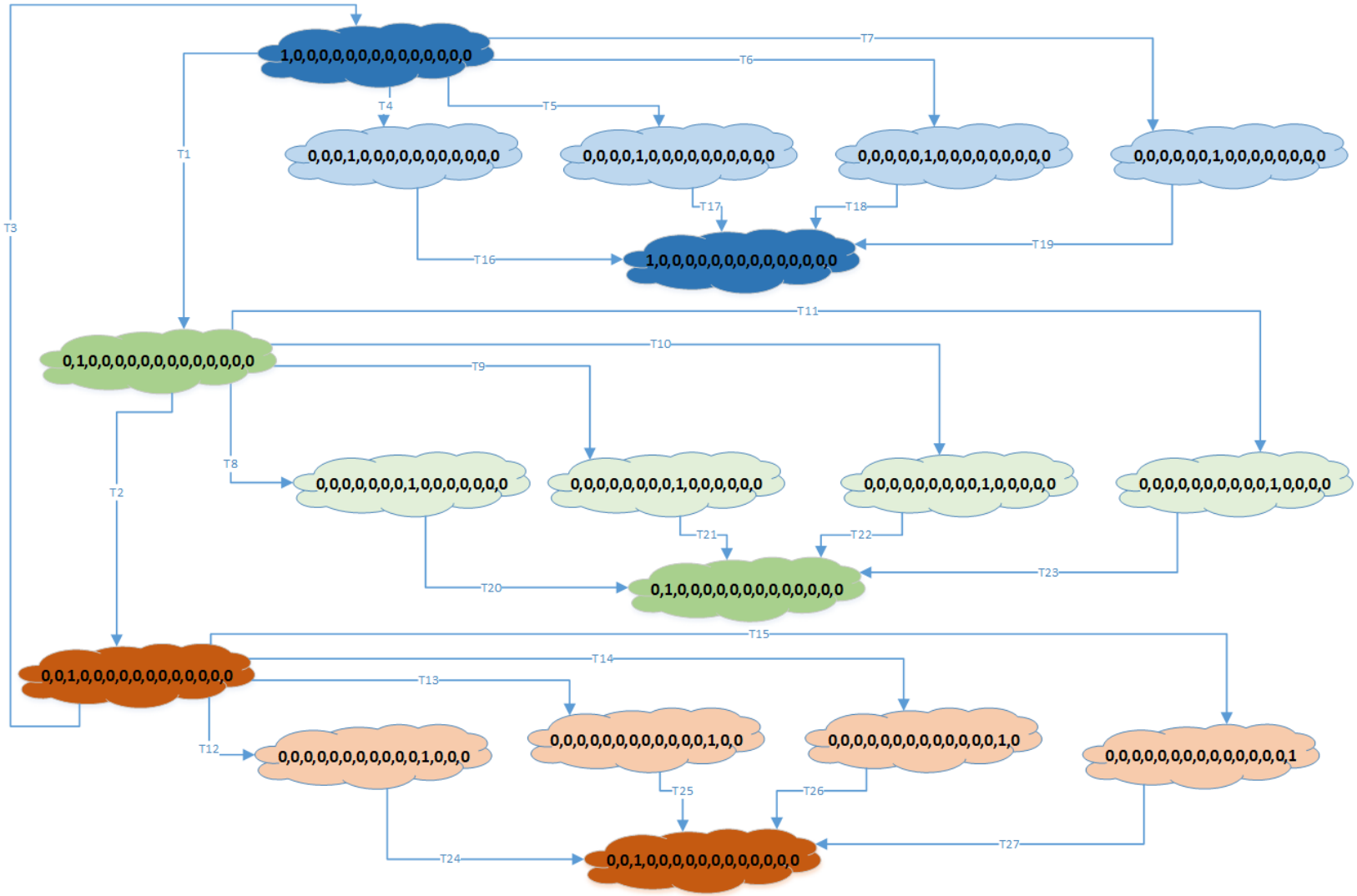
Anexos

Anexo 1 - Incidence Matrix

$$D = \begin{array}{c} \\ P1 \\ P2 \\ P3 \\ P4 \\ P5 \\ P6 \\ P7 \\ P8 \\ P9 \\ P10 \\ P11 \\ P12 \\ P13 \\ P14 \\ P15 \end{array} \begin{bmatrix} T1 & T2 & T3 & T4 & T5 & T6 & T7 & T8 & T9 & T10 & T11 & T12 & T13 & T14 & T15 & T16 & T17 & T18 & T19 & T20 & T21 & T22 & T23 & T24 & T25 & T26 & T27 \\ -1 & 0 & +1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 & +1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ +1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 & +1 & +1 & 0 & 0 & 0 & 0 \\ 0 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 & +1 & +1 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Clique para Voltar

Anexo 2 – Coverability Graph



Clique para Voltar