Model Checking

María José Ibañez
Instituto de Investigación en Ingeniería de Aragón (I3A)
Dpto. de Informática e Ing. Sistemas
Universidad de Zaragoza

mjibanez@unizar.es

OUTLINE

1. **MODEL CHECKING**
2. **MODELOS**
3. **LÓGICA TEMPORAL**
4. **HERRAMIENTAS**

OUTLINE

1. **MODEL CHECKING**
2. **MODELOS**
3. **LÓGICA TEMPORAL**
4. **HERRAMIENTAS**

1. MODEL CHECKING

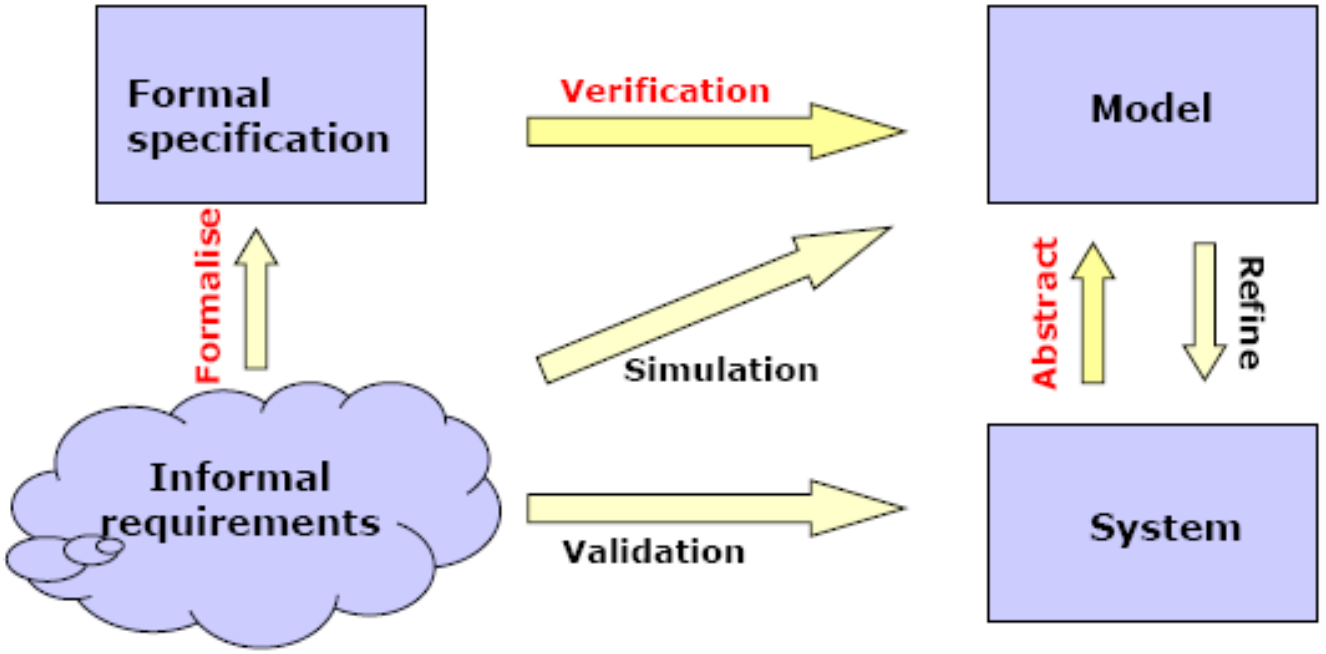
- Model checking es una **técnica de verificación automática** para sistemas concurrentes finitos.

- Fue desarrolada por **Clarke y Emerson** a principios de los 80.

1. MODEL CHECKING

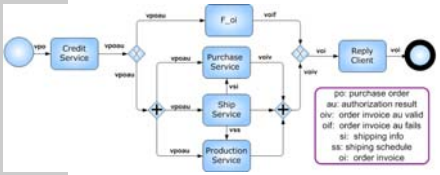
- Model checking es una **técnica de verificación automática** para sistemas concurrentes finitos.

- Fue desarrolada por **Clarke y Emerson** a principios de los 80.

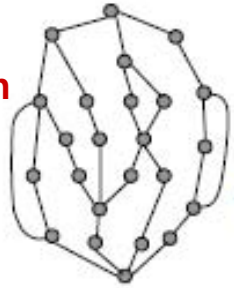


1. MODEL CHECKING

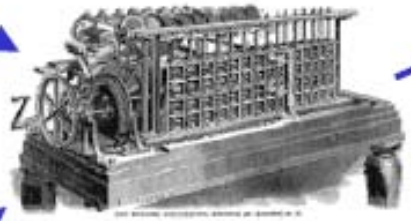
Sistema Real



Abstracción



Finite-state model



Model Checker

or

Error trace

```
Line 5: ...
Line 21: ...
Line 15: ...
  ...
Line 27: ...
Line 45: ...
```

Propiedades

Business Query 1
Business Query 2
Business Query 3
...

Formalización

$init \Rightarrow \mathbf{F}\ response$

Temporal logic specification

1. MODEL CHECKING

- Análisis de diversos tipos de sistemas
  - Sistemas en tiempo real
  - Sistemas híbridos
  - Programas escritos en Java o C
  - Aplicaciones Web
- Aplicaciones a varios dominios
  - Sistemas Industriales
  - Procesos de Negocio

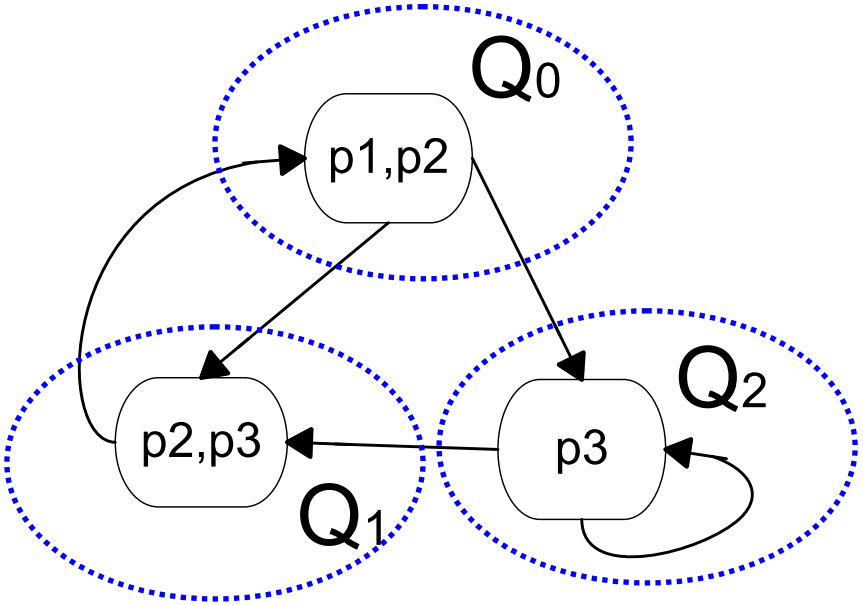
OUTLINE

1. MODEL CHECKING
2. MODELOS
3. LÓGICA TEMPORAL
4. HERRAMIENTAS

2. MODELOS

Una Estructura de Kripke M=(Q,T,L) consiste en

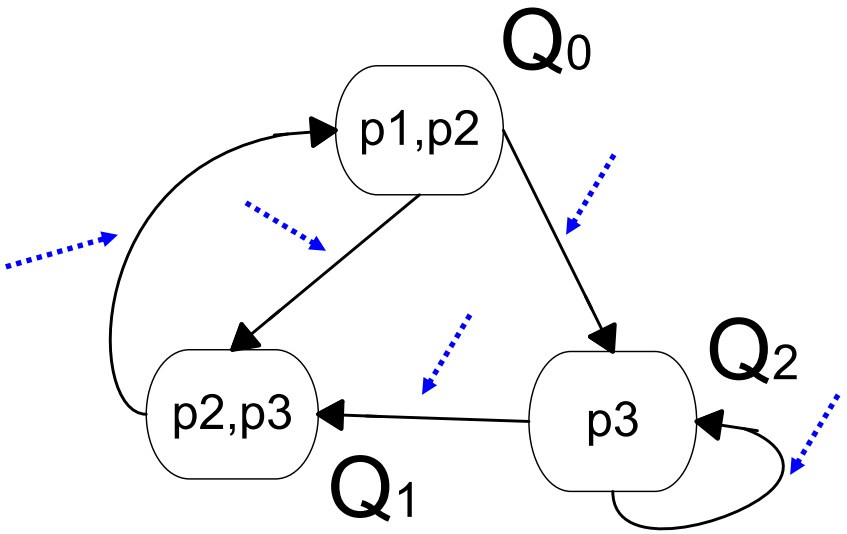
- Un conjunto de estados Q



2. MODELOS

Una Estructura de Kripke M=(Q,T,L) consiste en

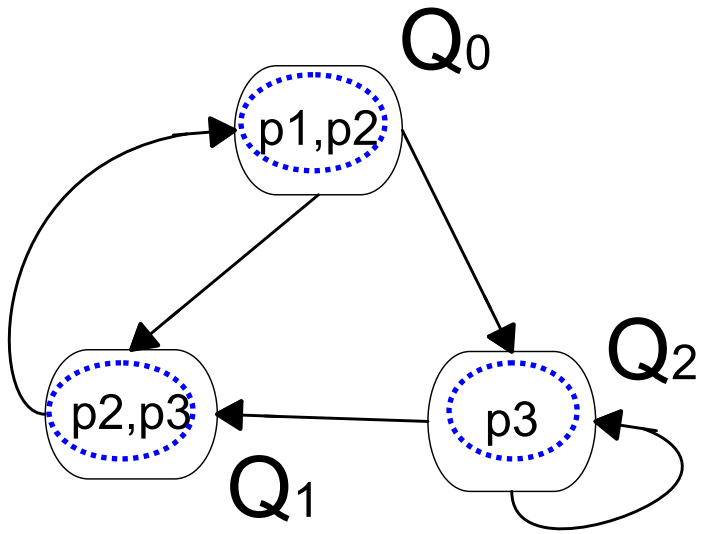
- Un conjunto de estados Q
- Un conjunto de transiciones T $\subseteq$ Q x Q (total relación)



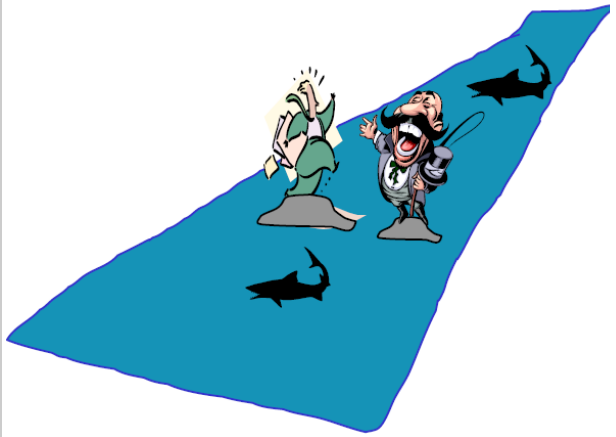
2. MODELOS

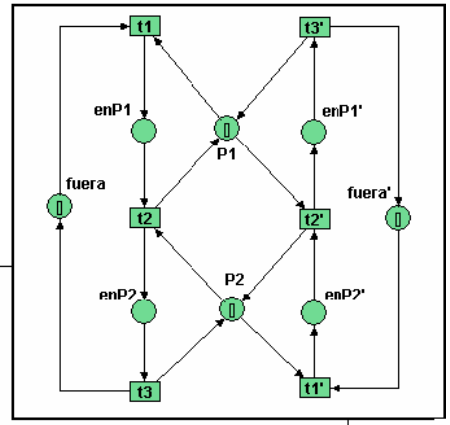
Una Estructura de Kripke M=(Q,T,L) consiste en

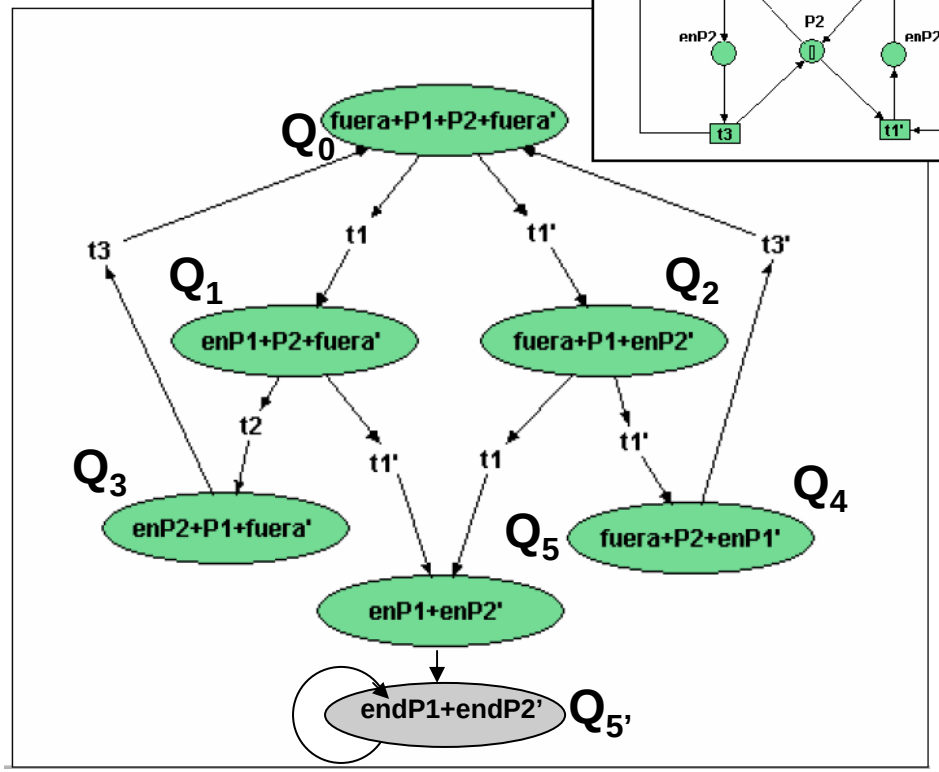
- Un conjunto de estados Q
- Un conjunto de transiciones $T \subseteq Q \times Q$ (total relación)
- Una función de Etiquetado $L : Q \rightarrow 2^{AP}$ con AP un conjunto de proposiciones atómicas.



2. MODELOS




t1
t3'
enP1
enP1'
P1
fuera
fuera'
t2
t2'
P2
enP2
enP2'
t3
t1'


Q0
fuera+P1+P2+fuera'
t1
t1'
t3
t3'
Q1
Q2
enP1+P2+fuera'
fuera+P1+enP2'
t2
t1'
t1
t1'
Q3
Q4
Q5
enP2+P1+fuera'
fuera+P2+enP1'
enP1+enP2'
endP1+endP2'
Q5'

OUTLINE

1. MODEL CHECKING
2. MODELOS
3. LÓGICA TEMPORAL
4. HERRAMIENTAS

3. LÓGICA TEMPORAL

La lógica temporal es usada para describir cualquier sistema de reglas y simbolismos para representar y razonar sobre proposiciones en términos de tiempo.

3. LÓGICA TEMPORAL

La lógica temporal es usada para describir cualquier sistema de reglas y simbolismos para representar y razonar sobre proposiciones en términos de tiempo.

Lógica No Temporal → “Tengo hambre." Tiempo constante

Lógica Temporal → “Siempre tengo hambre“ Tiempo no constante
“Eventualmente tendré hambre“
“Tendré hambre hasta que coma algo"

3. LÓGICA TEMPORAL

La lógica temporal es usada para describir cualquier sistema de reglas y simbolismos para representar y razonar sobre proposiciones en términos de tiempo.

Lógica No Temporal → "Tengo hambre." Tiempo constante

Lógica Temporal → "Siempre tengo hambre" / "Eventualmente tendré hambre" / "Tendré hambre hasta que coma algo" — Tiempo no constante

Lógica Temporal Lineal → Habilidad para razonar sobre una línea de tiempo.

Lógica temporal Ramificada → Habilidad para razonar sobre múltiples líneas de tiempo.

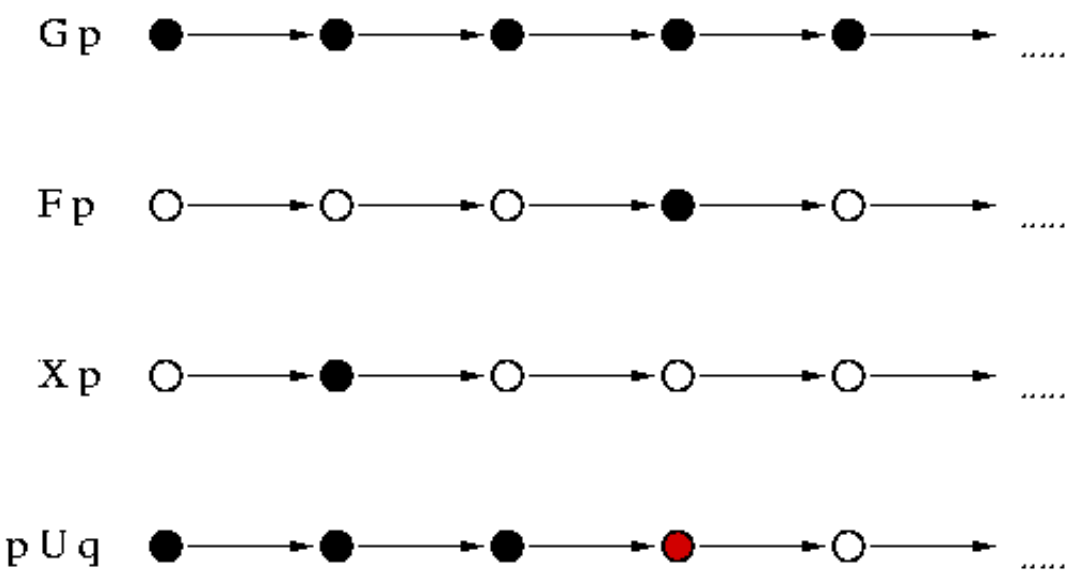
3. LÓGICA TEMPORAL

ELEMENTOS COMUNES CON OTRAS LÓGICAS

- Proposiciones atómicas: Afirmaciones sobre las ejecuciones del sistema.
- Operadores booleanos: Negación ($\neg$), Conjunción ($\wedge$), Disyunción ($\vee$) e implicación ($\Rightarrow$).
- Por ejemplo $a \wedge b \Rightarrow c$.
  - a, b, c son proposiciones atómicas.
  - $\wedge$, $\Rightarrow$ son operadores booleanos.

3. LÓGICA TEMPORAL -LTL Linear Temporal Logic

- Los operadores temporales expresan afirmaciones sobre el order de los eventos de una secuencia.
  - G, always (globalmente)
  - F. future (en el futuro)
  - X, next (siguiente)
  - U, until (hasta)



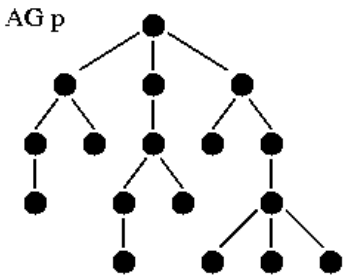
3. LÓGICA TEMPORAL -LTL Linear Temporal Logic

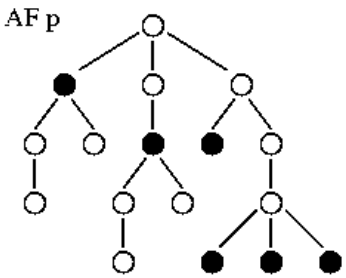
- Propiedades de seguridad, nada malo ocurrira:
  - $G \neg p$
- Propiedades de vivacidad: algo bueno continua ocurriendo.
  - GFp
  - $G \Rightarrow Fp$.
- Propiedades de equidad:
  - $GFp1 \Rightarrow GFp2$.

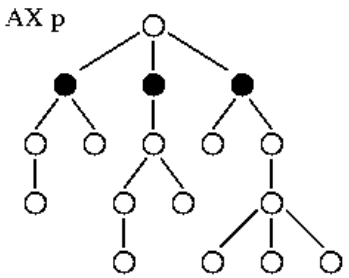
- Propiedades de exclusión mutua:
  - $G(\neg (pc1=12 \wedge pc2=22))$.

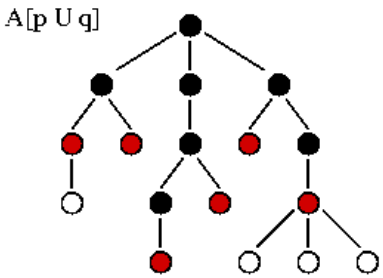
3. LÓGICA TEMPORAL - CTL Computational Tree Logic

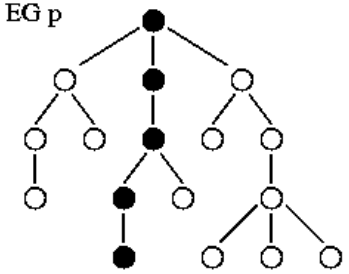
- Cada operador (G, F, X, U) es precedido por
  - A, along All paths (Siempre)
  - E, along at least (there Exists) one path (Existe)

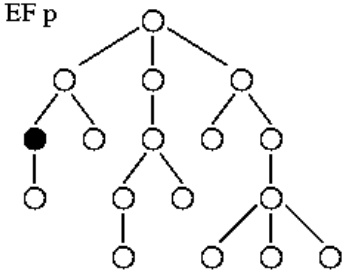


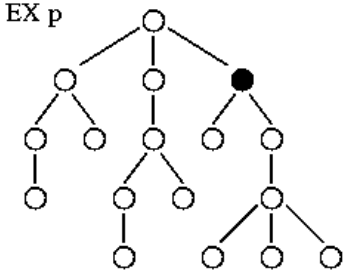


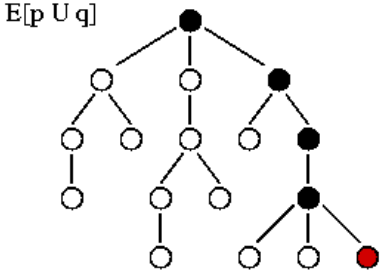




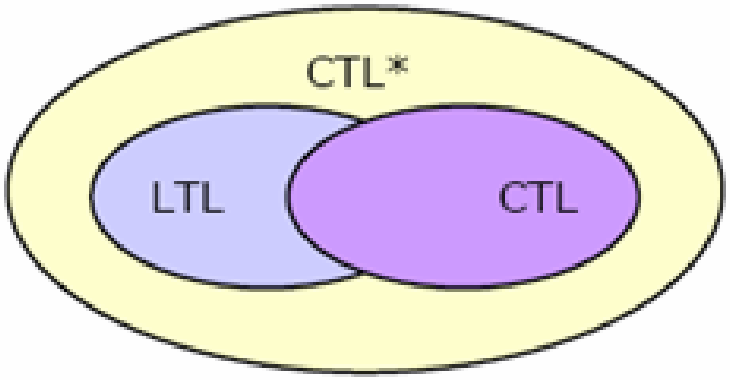








3. LÓGICA TEMPORAL - CTL ~ LTL ~ CTL*



FG p se puede expresar con LTL pero no con CTL
EF p se puede expresar con CTL pero no con LTL

CTL* es más expresiva pero los algoritmos son menos eficientes

¿Expresividad o Eficiencia?

OUTLINE

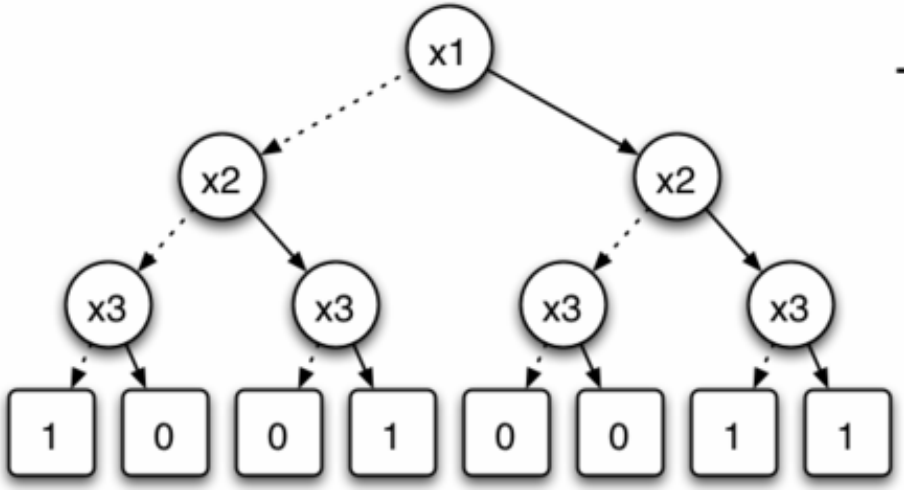
1. MODEL CHECKING
2. MODELOS
3. LÓGICA TEMPORAL
4. HERRAMIENTAS

4. HERRAMIENTAS DE MODEL CHECKING

- Hoy en día hay muchas herramientas disponibles para realizar model checking.
- Seleccionar una herramienta para un trabajo concreto es un trabajo difícil.
- Muchas de estas herramientas se basan en Binary Decision Diagrams (BDD)

4. HERRAMIENTAS DE MODEL CHECKING

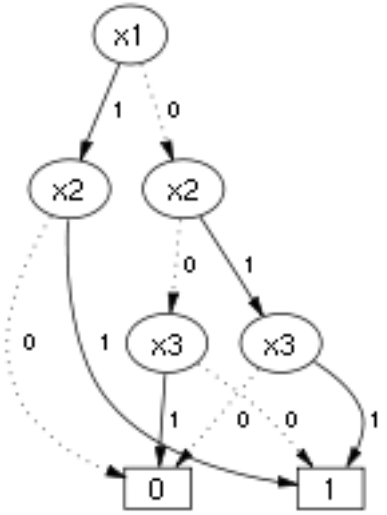
Los BDDs son una forma canónica y eficiente de representar funciones booleanas.



Binary Decision Tree

x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Truth Table



Binary Decision Diagram

4. HERRAMIENTAS DE MODEL CHECKING

- SPIN
  - Formalismo de Modelado: Promela. Los sistemas son vistos como un conjunto finito de máquinas de estado.
  - Model Checkers: on-the-fly LTL
- NUSMV: NuSMV es una reimplementación y extensión de SMV. Es el primer model checker basado en BDDs. Es usado en verificación industrial.
  - Formalismo de Modelado: SMV. Lenguaje de descripcin de circuitos.
  - Model Checkers: CTL y LTL model checkers.

4. HERRAMIENTAS DE MODEL CHECKING

- UPPAAL
  - Formalismo de Modelado: Sistemas de tiempo real. Autómata con tiempo.
  - Model Checkers: Algoritmo en C sobre el espacio de estados. CTL.
- KRONOS
  - Formalismo de Modelado: Sistemas de tiempo real. Autómata con tiempo.
  - Model Checkers: CTL con tiempo.
- HYTECH
  - Formalismo de Modelado: Autómata linear hibrido.
  - Model Checkers: Conjunto de instrucciones de control.

4. HERRAMIENTAS DE MODEL CHECKING

LOLA (a Low Level Petri Net Analyzer)

- Ha sido implementado para validar las técnicas de reducción a partir de su grafo de alcanzabilidad. Puede analizar alcanzabilidad, acotación, bloqueos, transiciones muertas y reversibilidad.
- LoLA puede analizar las siguientes propiedades en una red de Petri:
  - Reversibilidad de un estado dado.
  - Limitación de un lugar y una red.
  - Cuasivivacidad de una transición dada.
  - Reversibilidad de una red.
  - Existencia de estados home.
- Además puede hacer Model checking para CTL
- Todos los problemas son verificados recorriendo el espacio de estados. Se utilizan las siguientes técnicas:
  - Reducción simétrica.
  - Conjunto de reducciones simples.
  - Grafo de cobertura reducido.
- No soporta redes de Petri de alto nivel.

4. HERRAMIENTAS DE MODEL CHECKING

PEP Tool (Programming Environment based on Petri Nets)

- Es un extenso conjunto de componentes de modelados, compilación, simulación y verificación junto con una interfaz gráfica Tcl-Tk. Algunas de sus características son las siguientes:
  - Modela fácilmente diseños de sistemas paralelos, autómatas finitos, algebras de procesos y redes de Petri de alto nivel.
  - Compila redes de Petri de estos modelos.
  - Verifica propiedades de alcanzabilidad, bloqueos y verificación de algoritmos.
  - Soporta los analizadores de INA, FC2Tools, SMV, LOLA, PROD, DSSZ y SPIN.
  - Posee interfaz gráfica.
  - Se pueden incluir algoritmos fácilmente en ella.
- Solo verifica formulas de lógica temporal sobre redes ordinarias.

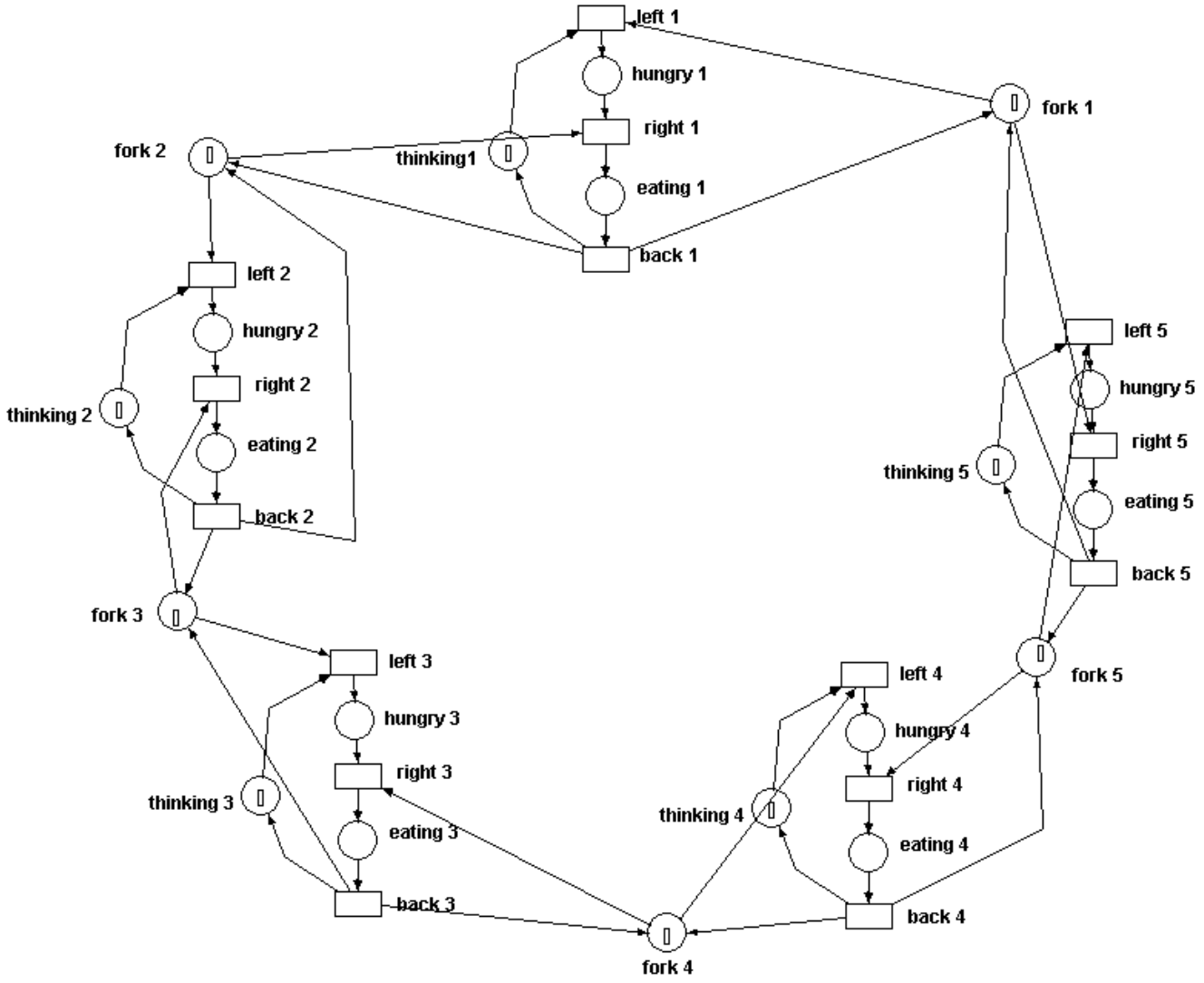
4. HERRAMIENTAS DE MODEL CHECKING

MARIA

- Se trata de una herramienta sucesora de PROD. En un análisis automático detecta bloqueos y errores en computación.
  - Admite redes de Petri de alto nivel.
  - Se trata de un on-the-fly LTL model checking.
- http://www.tcs.hut.fi/Software/maria/index.en.html

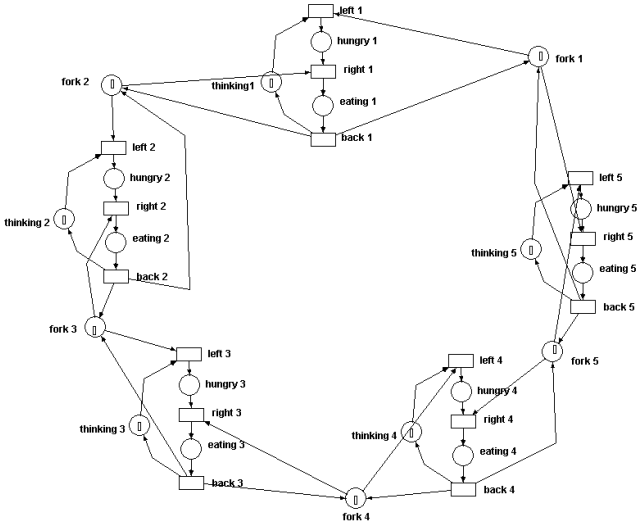
4. HERRAMIENTAS DE MODEL CHECKING

PROBLEMA DE LOS FILOSOFOS 1



4. HERRAMIENTAS DE MODEL CHECKING

PROBLEMA DE LOS FILOSOFOS 1

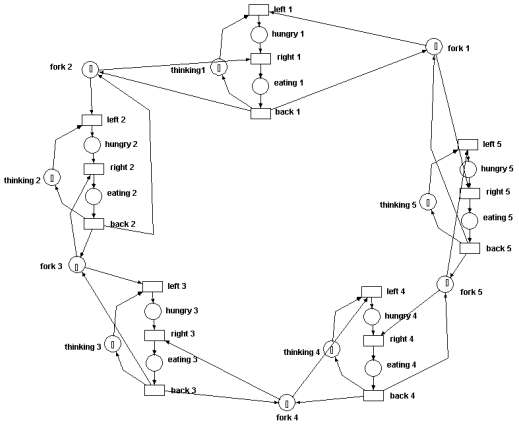



```
typedef unsigned (1..1) phil_t;
typedef unsigned (1..2) token;

place fork1 (0..#phil_t) phil_t: 1#1;
place thinking1  (0..#phil_t) phil_t: 1#1;
place hungry1  (0..#phil_t) phil_t;
place eating1  (0..#phil_t) phil_t;
…
```

```
trans left1
in { place fork1: p;  place thinking1: p; }
out { place hungry1: p; };

trans right1
in { place fork2: p;  place hungry1: p;}
out {  place eating1: p; };

trans finish1
in { place eating1: p;  }
out { place thinking1: p;  place fork1: p;
place fork2: p; };

…
deadlock true;
```

4. HERRAMIENTAS DE MODEL CHECKING

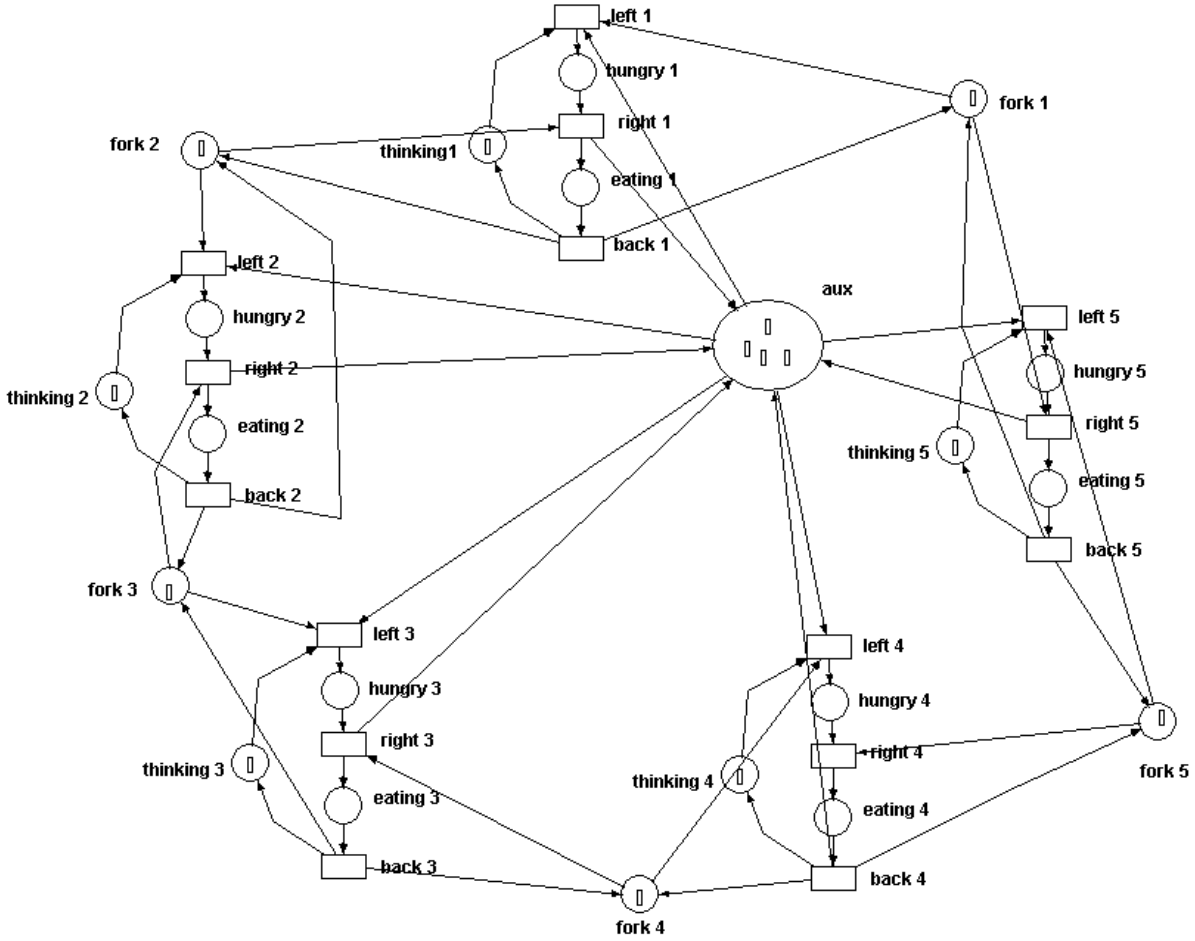
PROBLEMA DE LOS FILOSOFOS 1




```
C:\maria>maria -b dining1.pn
deadlock state @78
"dining1.pn": 82 states (3 bytes), 1 error, 265 arcs
@0$show @78
@78:deadlock state (
 hungry1:
  1
 hungry2:
  1
 hungry3:
  1
 hungry4:
  1
 hungry5:
  1
)
5 predecessors
@0$_
```

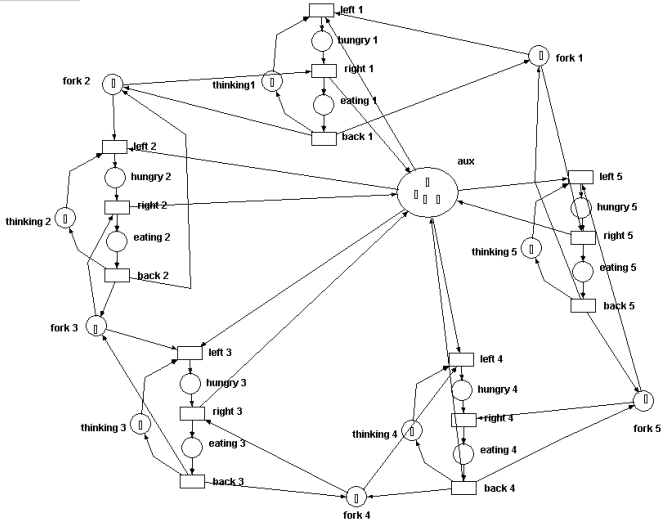
4. HERRAMIENTAS DE MODEL CHECKING

PROBLEMA DE LOS FILOSOFOS 1



4. HERRAMIENTAS DE MODEL CHECKING

PROBLEMA DE LOS FILOSOFOS 1

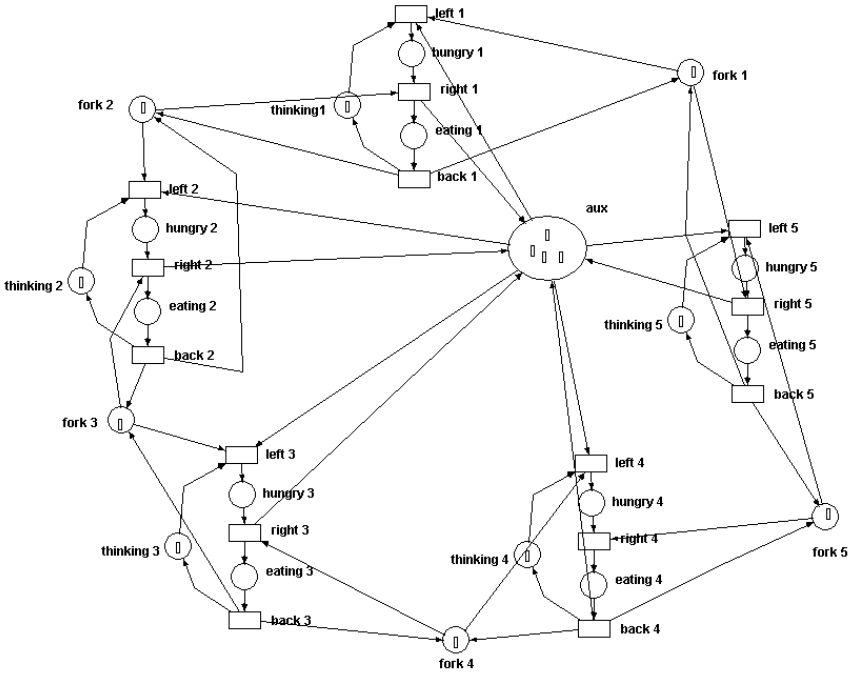



```
typedef unsigned (1..1) phil_t;
typedef unsigned (1..4) token;

place fork1 (0..#phil_t) phil_t: 1#1;
place thinking1  (0..#phil_t) phil_t: 1#1;
place hungry1  (0..#phil_t) phil_t;
place eating1  (0..#phil_t) phil_t;
…
place aux (0..#token) token: 4#1;
```

```
trans left1
in { place fork1: p;  place thinking1: p;
place aux: 1;}
out { place hungry1: p; };

trans right1
in { place fork2: p;  place hungry1: p;}
out {  place eating1: p; place aux: 1;};

trans finish1
in { place eating1: p;  }
out { place thinking1: p;  place fork1: p;
place fork2: p; };

…
deadlock true;
```

4. HERRAMIENTAS DE MODEL CHECKING

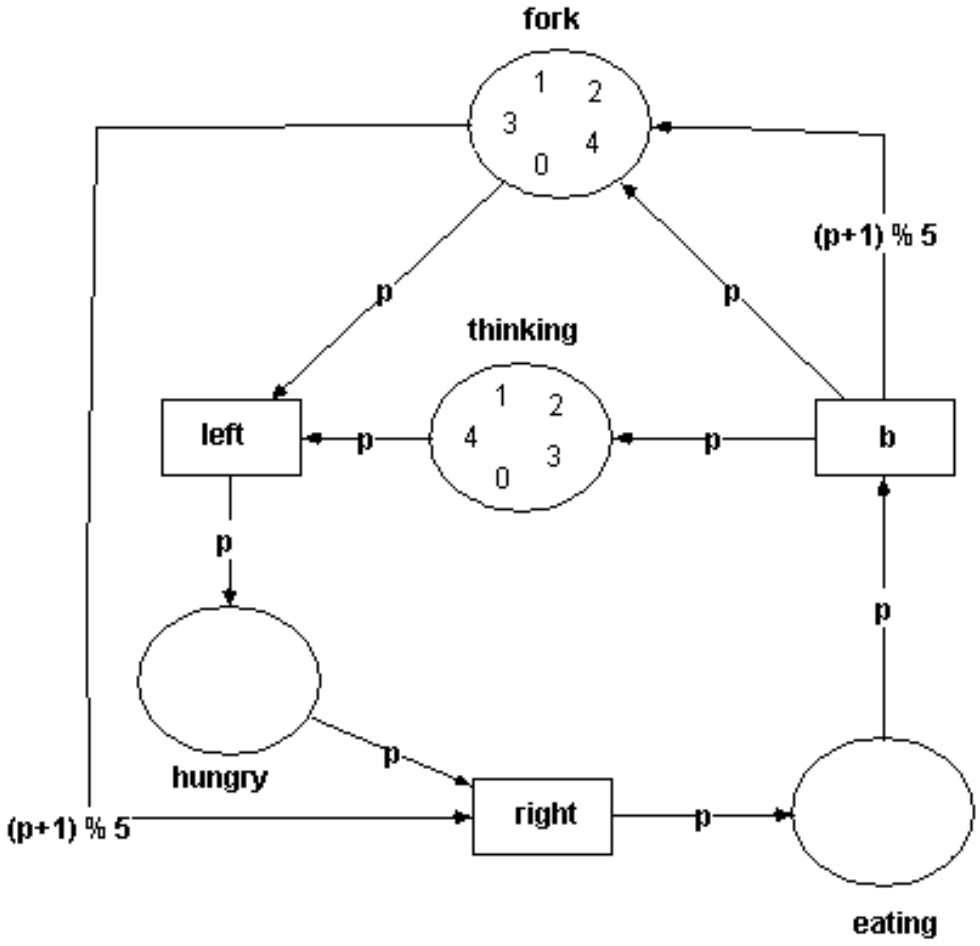
PROBLEMA DE LOS FILOSOFOS 1



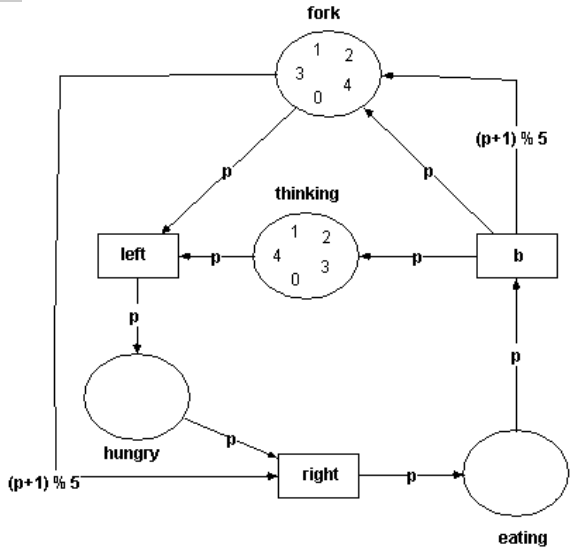

```
C:\maria>maria -b dining1NB.pn
"dining1NB.pn": 81 states (3..4 bytes), 260 arcs
C@$
```

4. HERRAMIENTAS DE MODEL CHECKING

PROBLEMA DE LOS FILOSOFOS 2



4. HERRAMIENTAS DE MODEL CHECKING
PROBLEMA DE LOS FILOSOFOS 2

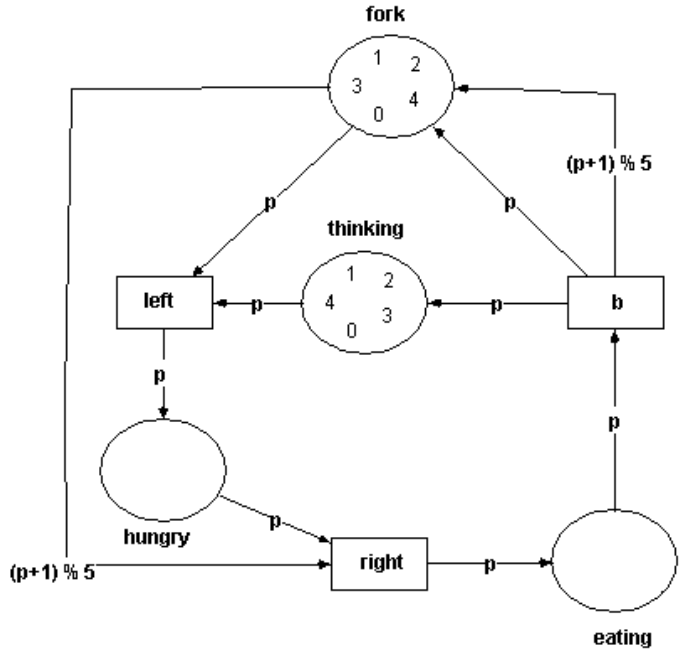



```
typedef unsigned (1..5) phil_t;

place fork (0..#phil_t) phil_t: phil_t p: p;

place thinking  (0..#phil_t) phil_t: phil_t p: p;

place hungry  (0..#phil_t) phil_t;

place eating  (0..#phil_t) phil_t;
```

```
trans left
in { place fork: p;  place thinking: p;}
out { place hungry: p; };

trans right
in { place fork: +p;  place hungry: p;}
out {  place eating: p; };

trans finish
in { place eating: p;  }
out { place thinking: p;  place fork: p, +p; };

deadlock true;
```

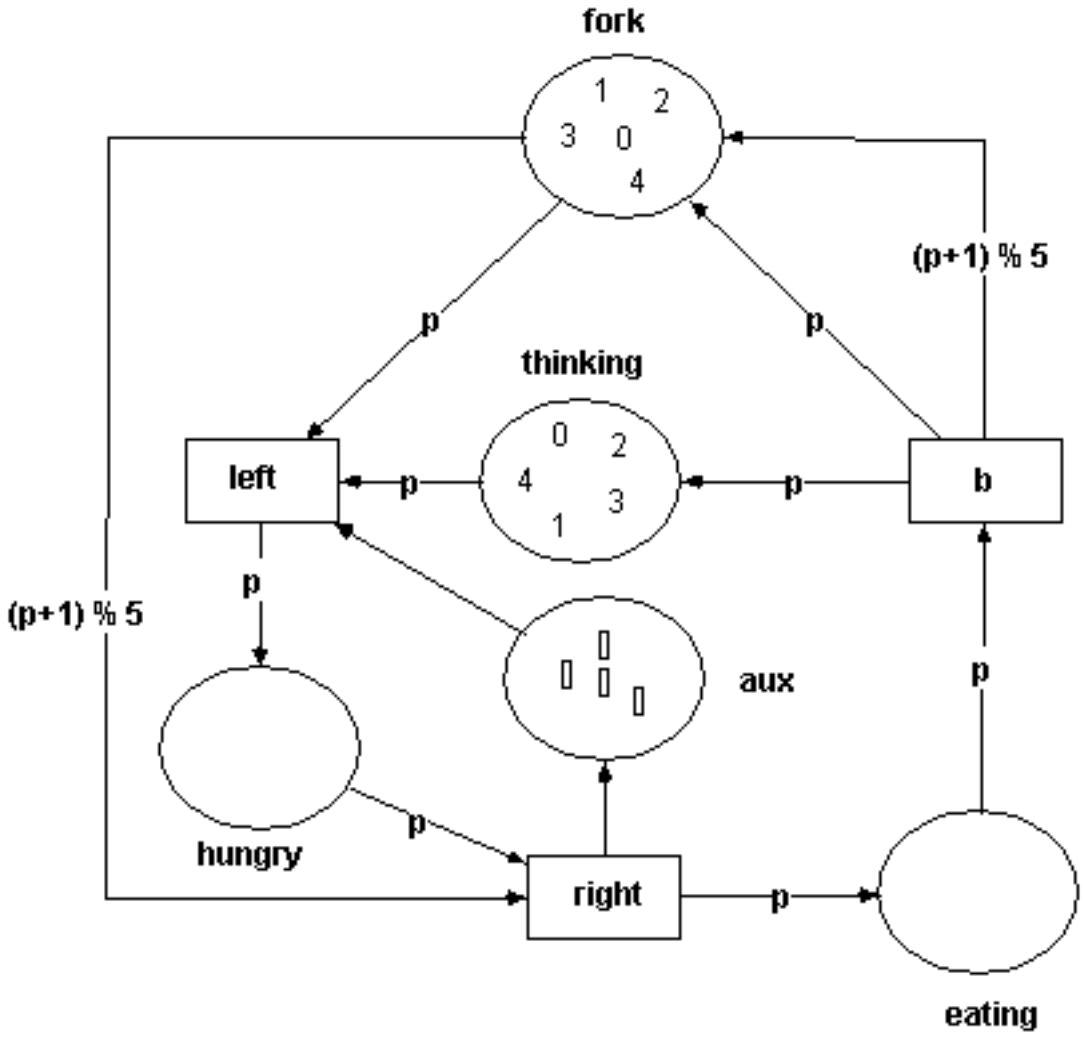
4. HERRAMIENTAS DE MODEL CHECKING

PROBLEMA DE LOS FILOSOFOS 2



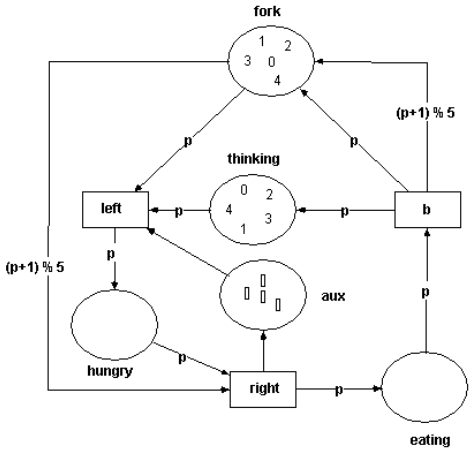

```
C:\maria>maria -b dining2.pn
deadlock state @71
"dining2.pn": 82 states (4..6 bytes), 1 error, 265 arcs
@0$show @71
@71:deadlock state (
 hungry:
  1,2,3,4,5
)
5 predecessors
@0$
```

4. HERRAMIENTAS DE MODEL CHECKING
PROBLEMA DE LOS FILOSOFOS 2



4. HERRAMIENTAS DE MODEL CHECKING

PROBLEMA DE LOS FILOSOFOS 2

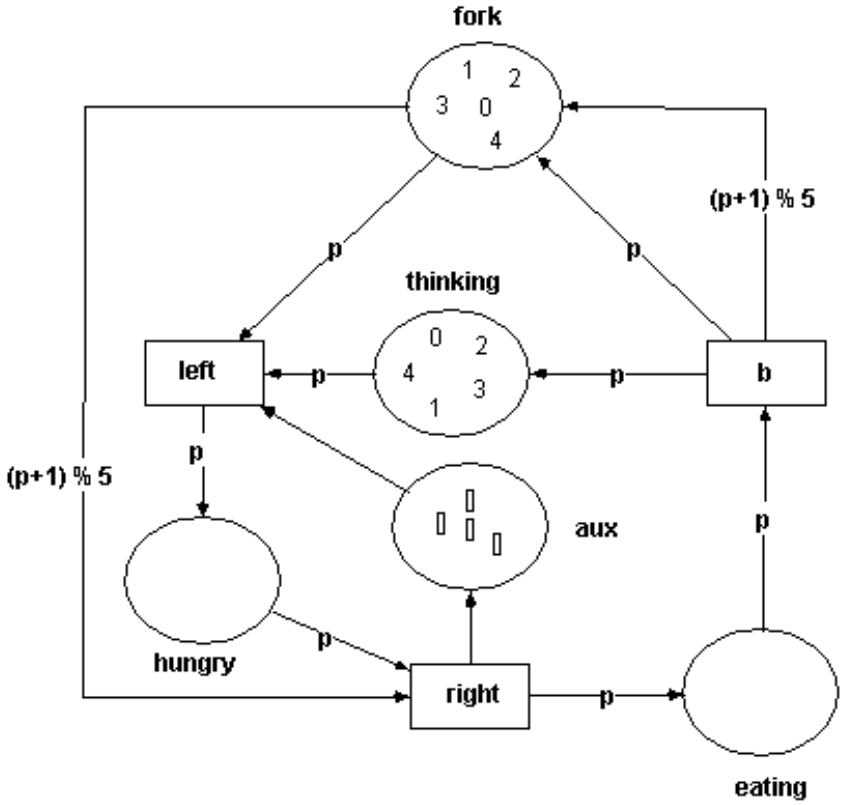



```
typedef unsigned (1..5) phil_t;
typedef unsigned (1..4) token;

place fork (0..#phil_t) phil_t: phil_t p: p;

place thinking  (0..#phil_t) phil_t: phil_t p: p;

place hungry  (0..#phil_t) phil_t;

place eating  (0..#phil_t) phil_t;

place aux (0..#token) token: 4#1;
```

```
trans left
in { place fork: p;  place thinking: p; place
aux: 1;}
out { place hungry: p; };

trans right
in { place fork: +p;  place hungry: p;}
out {  place eating: p; place aux: 1;};

trans finish
in { place eating: p;  }
out { place thinking: p;  place fork: p, +p; };

deadlock true;
```

4. HERRAMIENTAS DE MODEL CHECKING

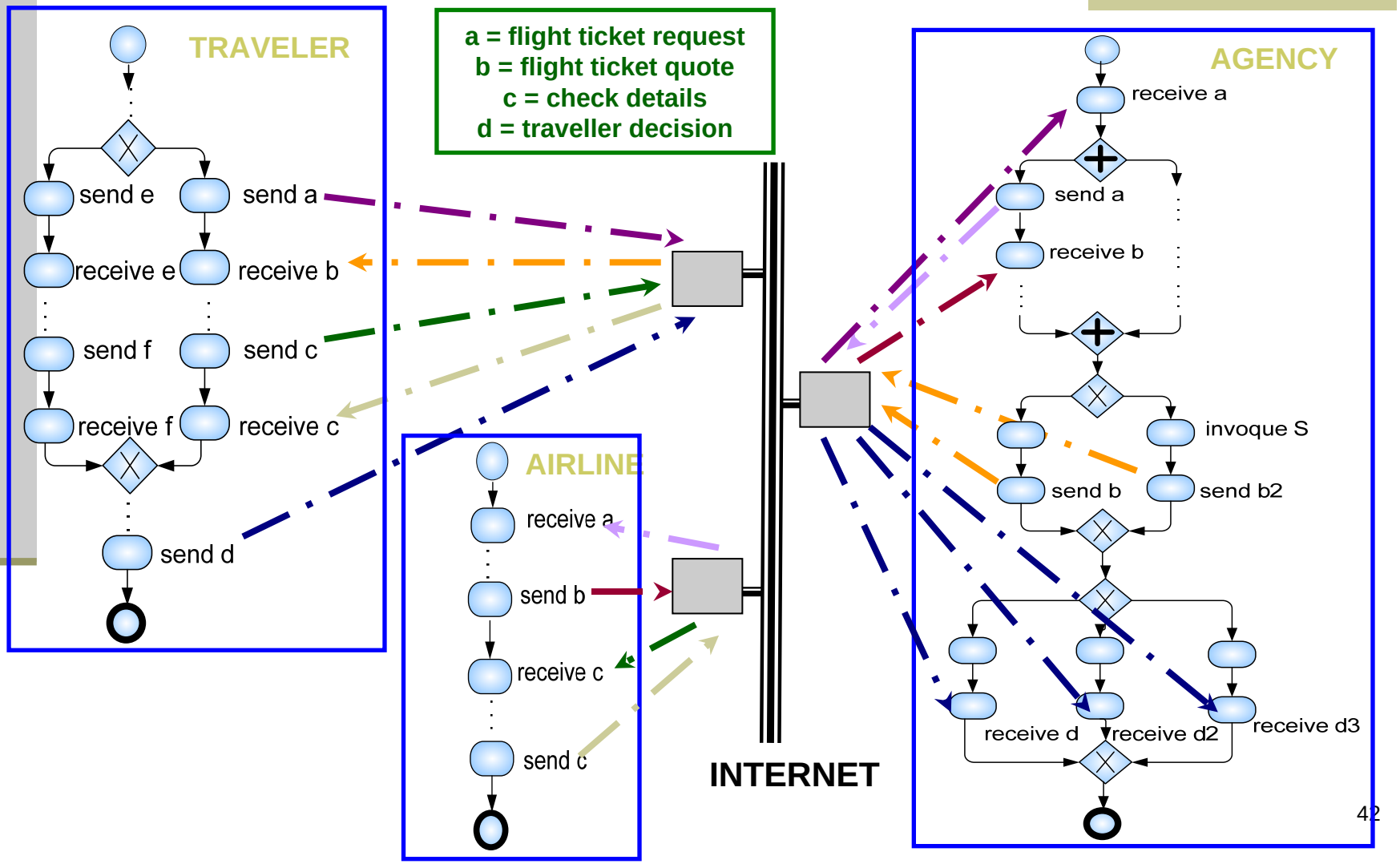
PROBLEMA DE LOS FILOSOFOS 2




```
C:\maria>maria -b dining2NB.pn
"dining2NB.pn": 81 states (5..7 bytes), 260 arcs
@0$
```

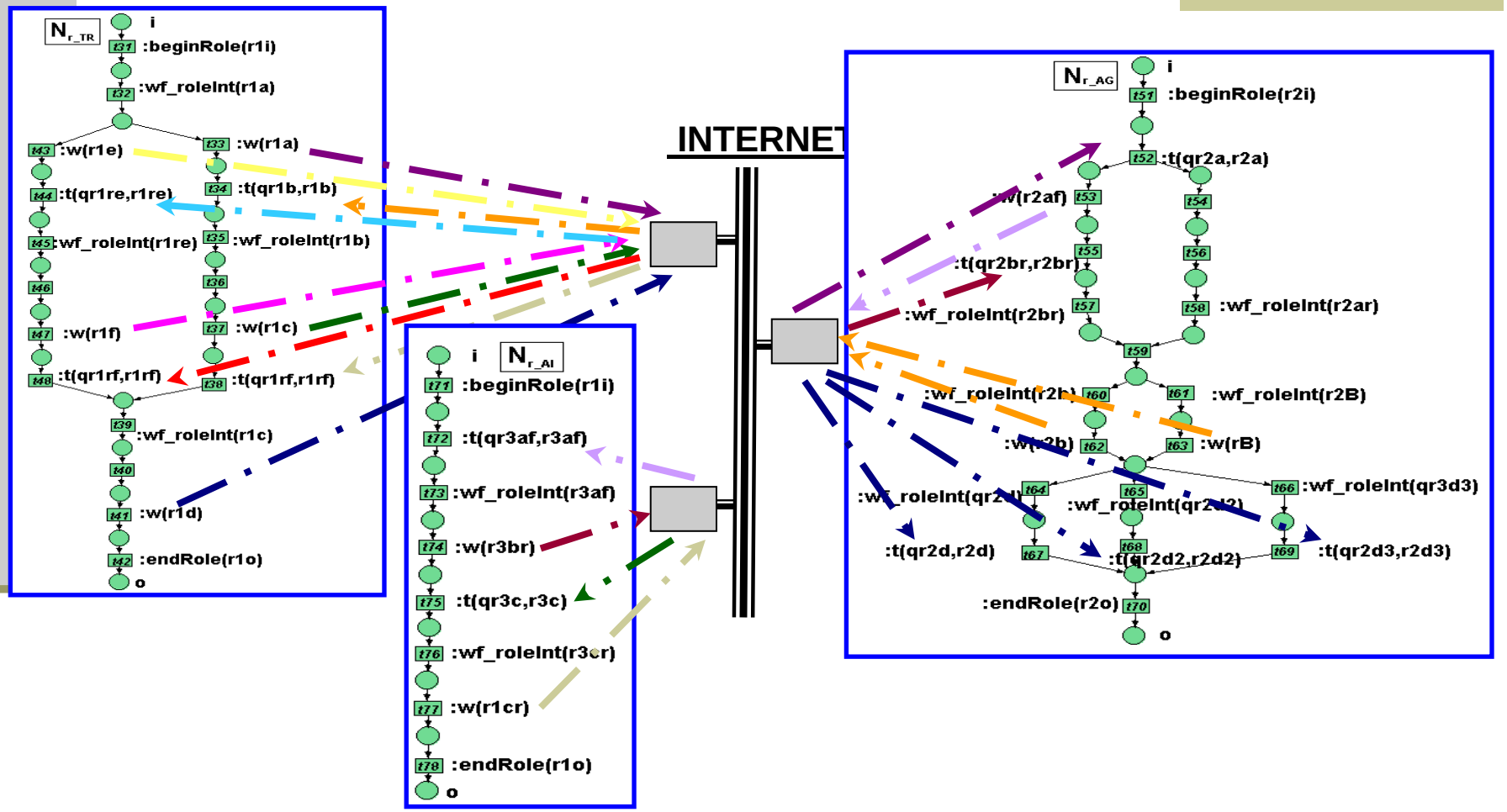
4. HERRAMIENTAS DE MODEL CHECKING

COMPATIBILIDAD DE PROCESOS WEB

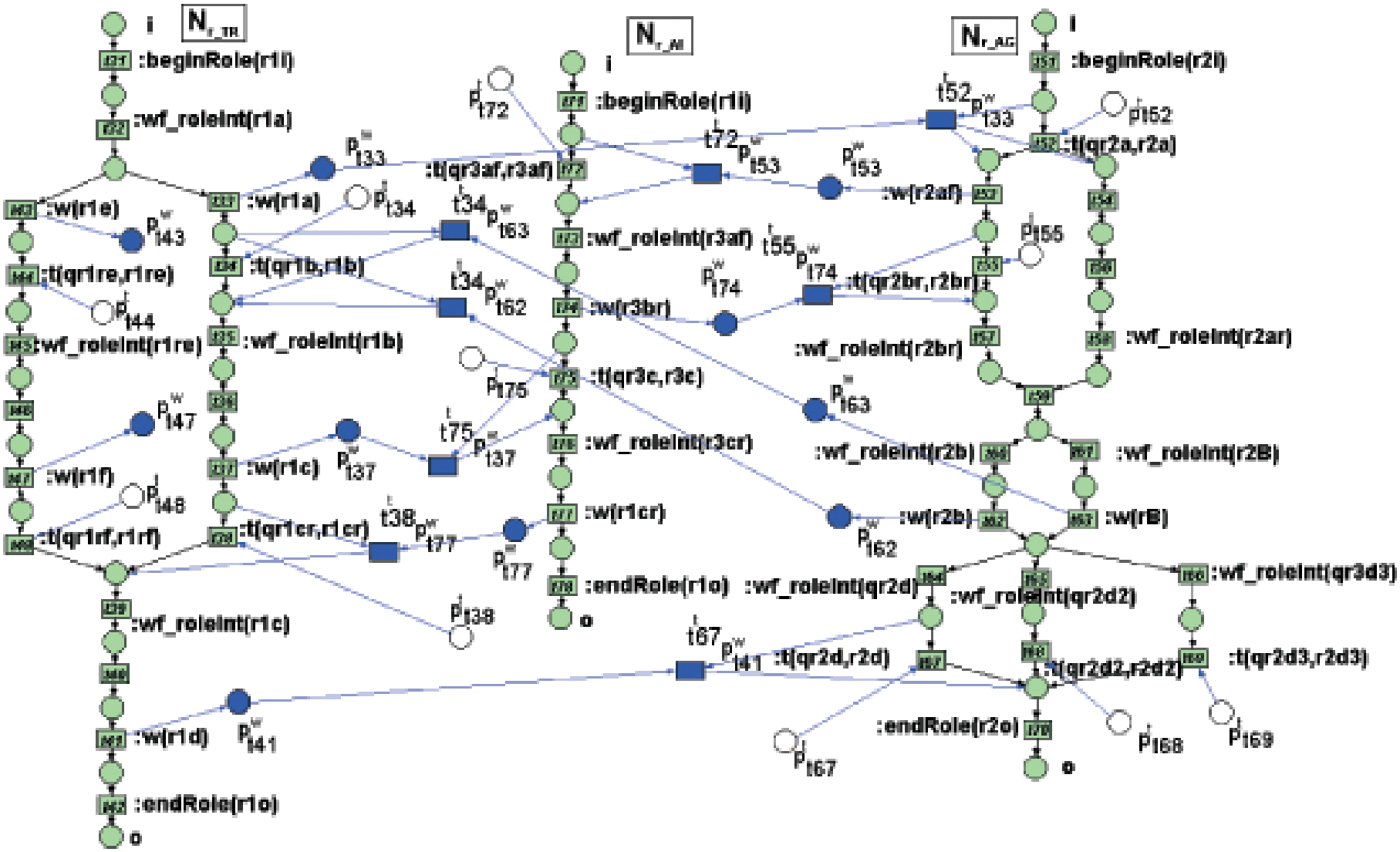


4. HERRAMIENTAS DE MODEL CHECKING

COMPATIBILIDAD DE PROCESOS WEB

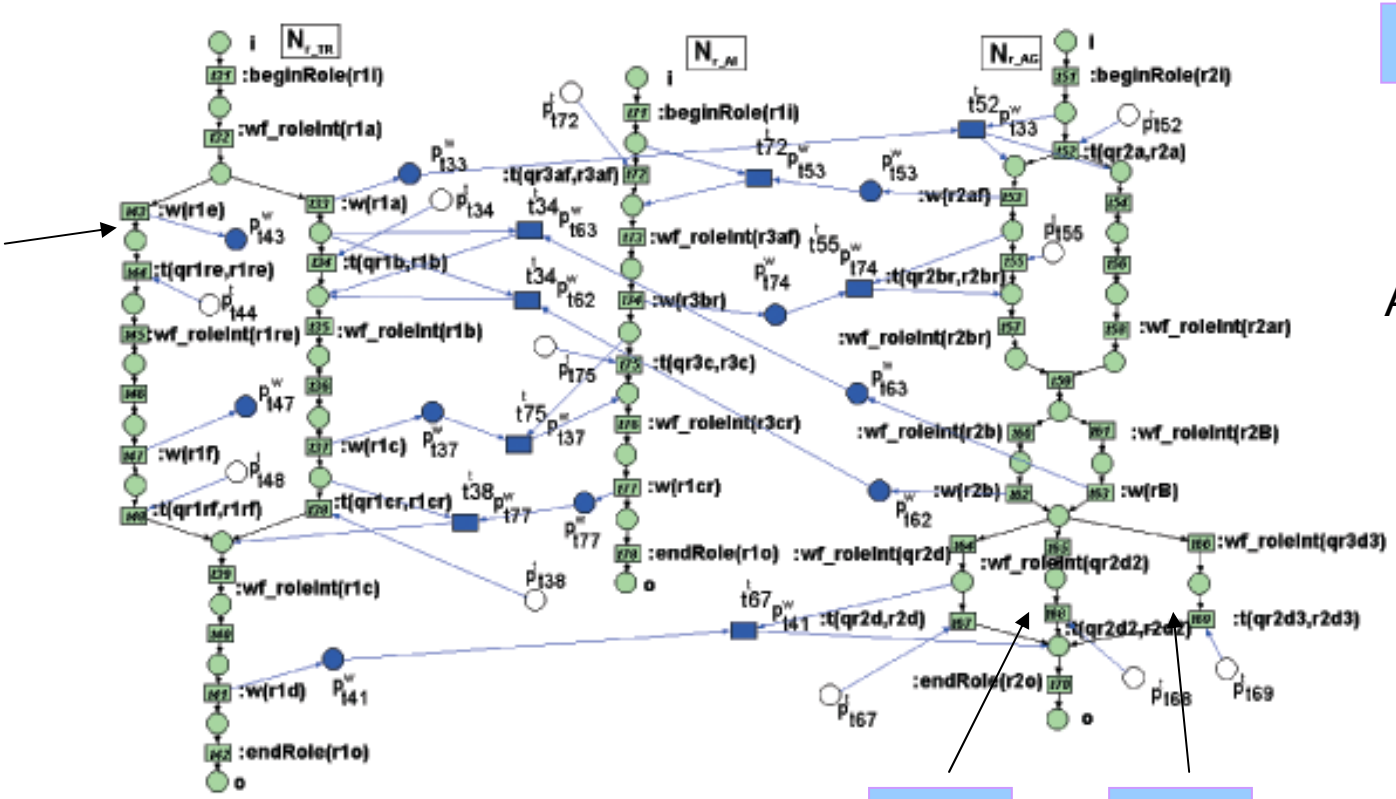


4. HERRAMIENTAS DE MODEL CHECKING
COMPATIBILIDAD DE PROCESOS WEB



4. HERRAMIENTAS DE MODEL CHECKING
COMPATIBILIDAD DE PROCESOS WEB

```
C:\maria>maria -b COMP.pn
deadlock state @18
deadlock state @138
deadlock state @139
deadlock state @146
"COMP.pn": 147 states (9 bytes), 4 errors, 278 arcs
@0$
```



@18

@146

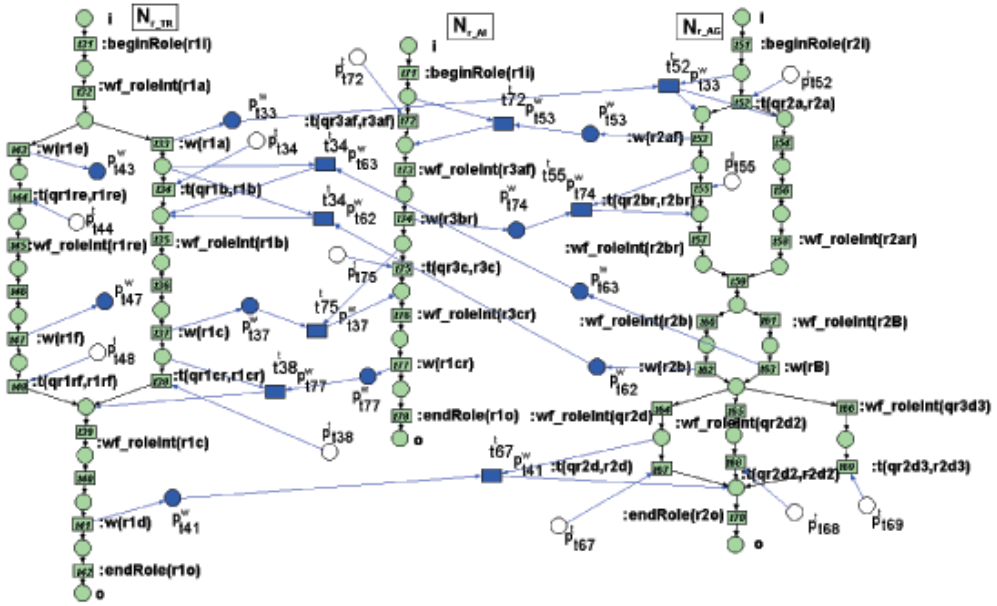
Acaba bién

@28

@28

4. HERRAMIENTAS DE MODEL CHECKING

COMPATIBILIDAD DE PROCESOS WEB

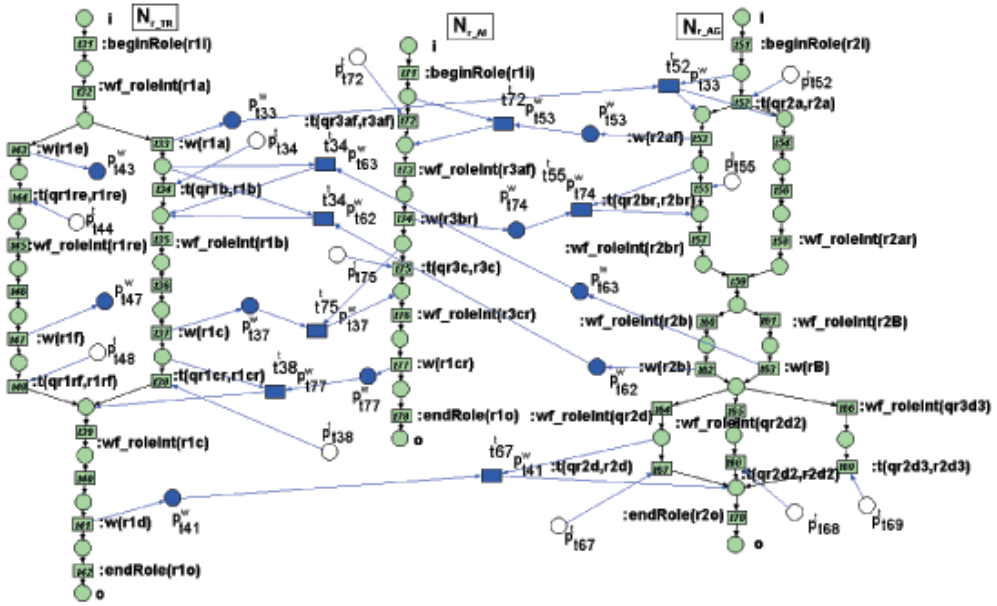


[](place P12 equals 1) and (place P47 equals 1) and (place P61 equals 1)

A partir de ahora se alcanzará un estado en el que los 3 procesos acaben. FALSO

4. HERRAMIENTAS DE MODEL CHECKING

COMPATIBILIDAD DE PROCESOS WEB

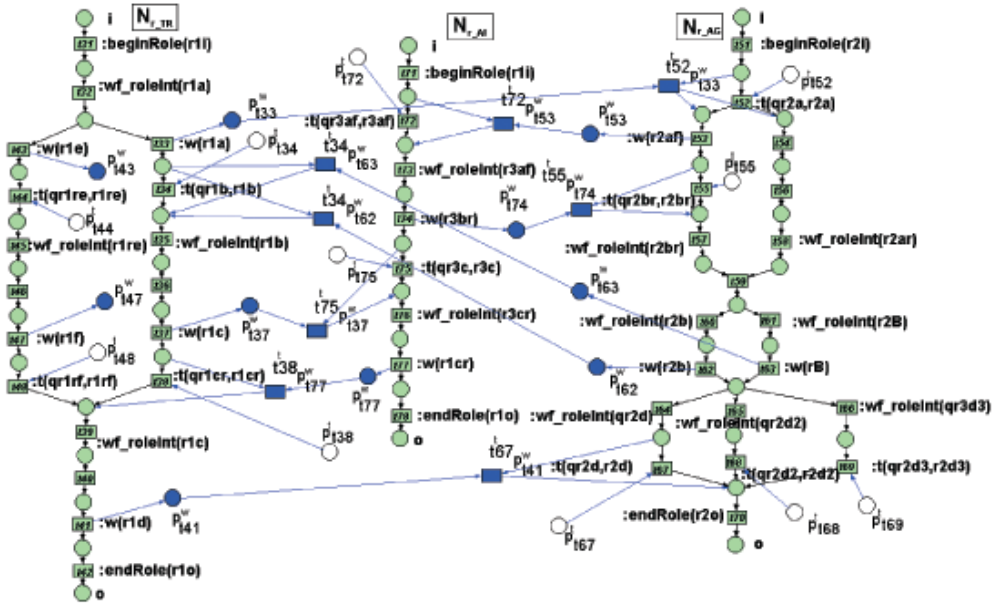


<>(place P12 equals 1) and (place P47 equals 1) and (place P61 equals 1)

A partir de ahora en todos los estados los 3 procesos han acabado.
FALSO

4. HERRAMIENTAS DE MODEL CHECKING

COMPATIBILIDAD DE PROCESOS WEB

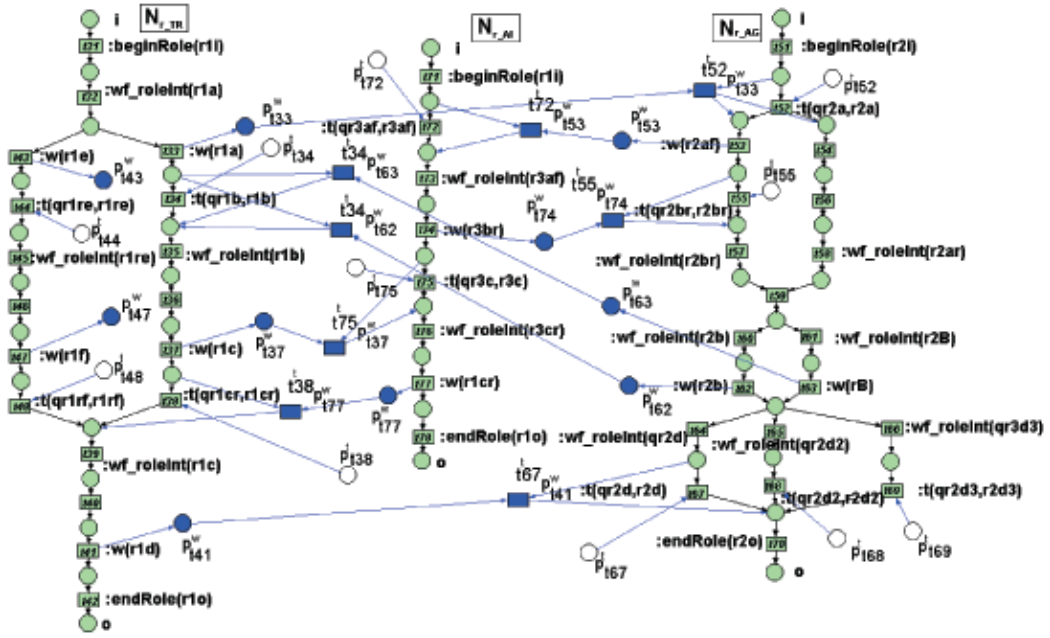


<>[](place P12 equals 1) and (place P47 equals 1) and (place P61 equals 1)

Tarde o temprano se alcanzará un estado en el que los 3 procesos acaben. FALSO

4. HERRAMIENTAS DE MODEL CHECKING

COMPATIBILIDAD DE PROCESOS WEB



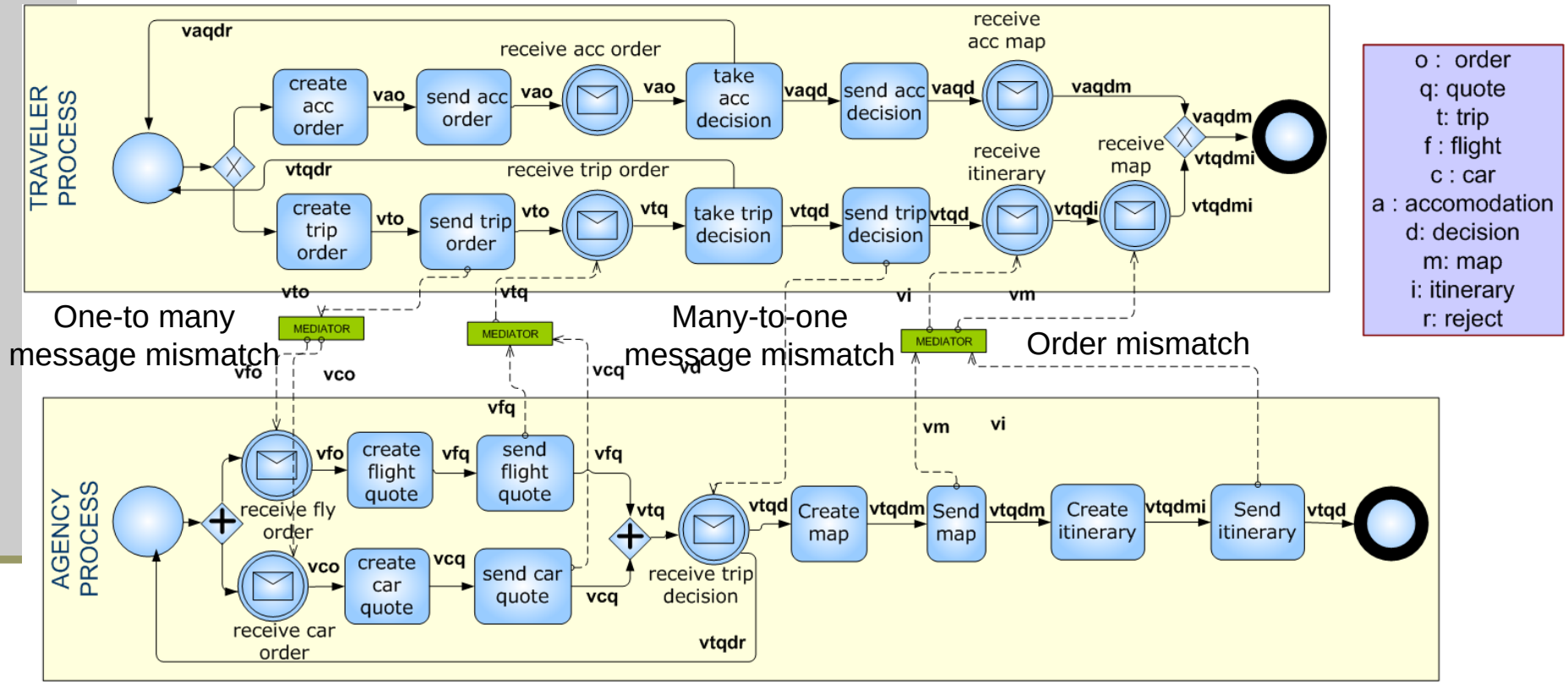

```
"COMP.pn": 147 states (9 bytes), 4 errors, 278 arcs
@0$<>[](place P12 equals 1) and (place P47 equals 1) and (place P61 equals 1)
(command line):1:constructing counterexample
(command line):1:property holds
```

<>[](place P12 equals 1) and (place P47 equals 1) and (place P61 equals 1)

A partir de un punto en el futuro en el que los 3 procesos han acabado, su estado no cambiará. VERDADERO

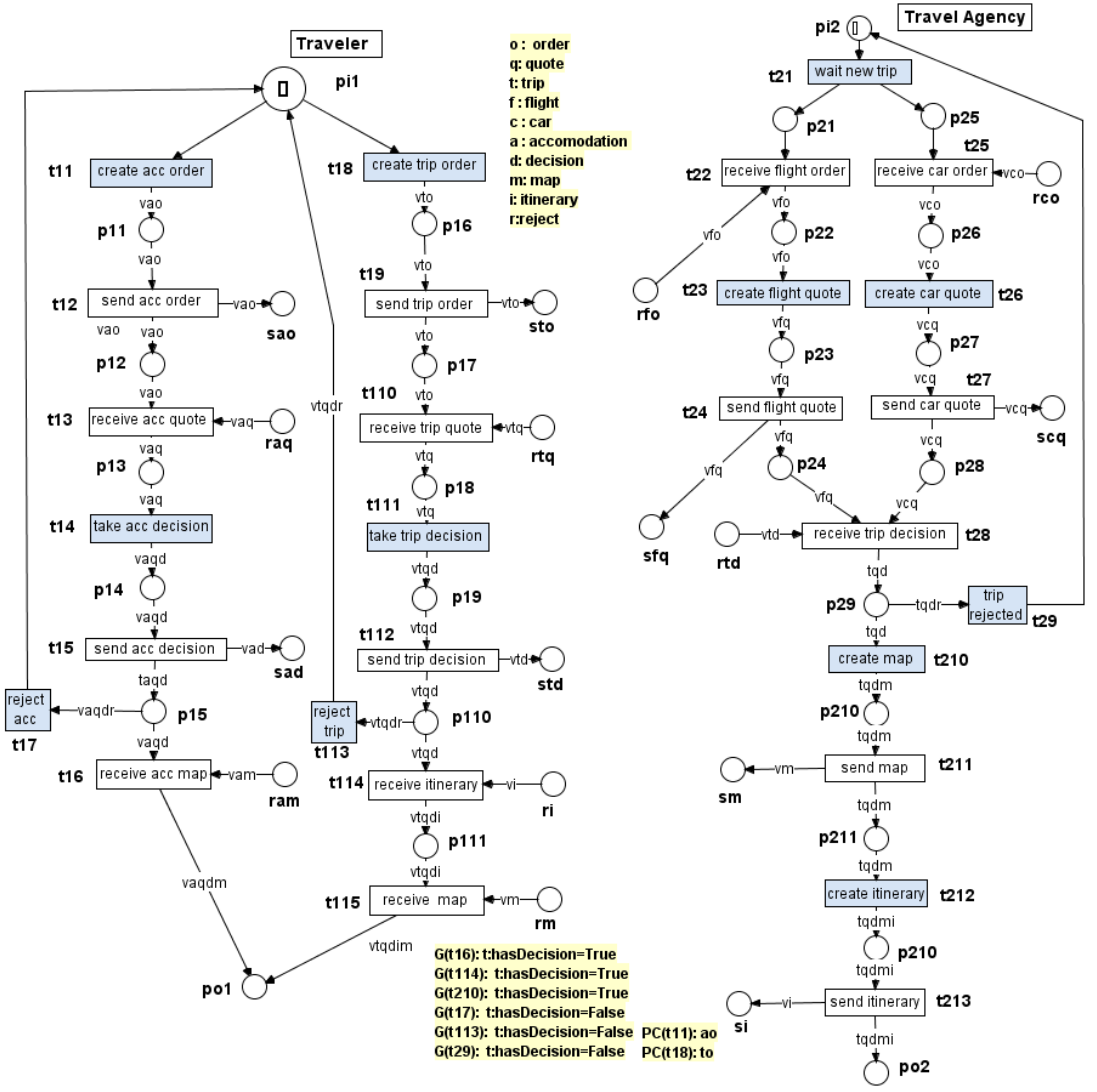
4. HERRAMIENTAS DE MODEL CHECKING

MEDIACIÓN DE PROCESOS WEB



4. HERRAMIENTAS DE MODEL CHECKING

MEDIACIÓN DE PROCESOS WEB



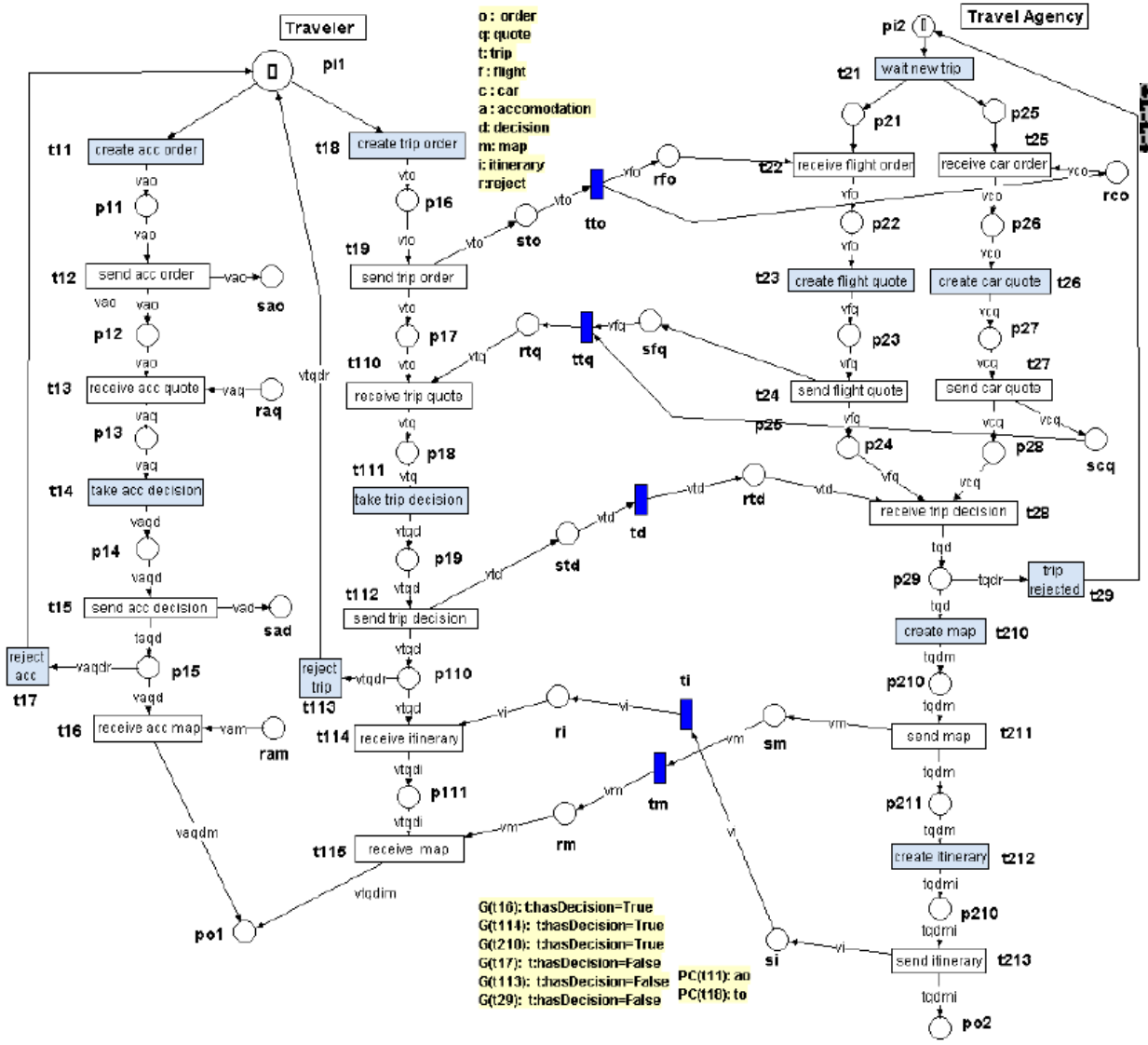

```
C:\maria>maria -b ExAdaptabilidad.pn
deadlock state @8
deadlock state @9
```

```
@8:deadlock state (
 p12:
  1
 sao:
  1
 p21:
  1
 p25:
  1
)
2 predecessors
```

```
@0$show @9
@9:deadlock state (
 p17:
  1
 sto:
  1
 p21:
  1
 p25:
  1
)
2 predecessors
```

4. HERRAMIENTAS DE MODEL CHECKING

MEDIACIÓN DE PROCESOS WEB

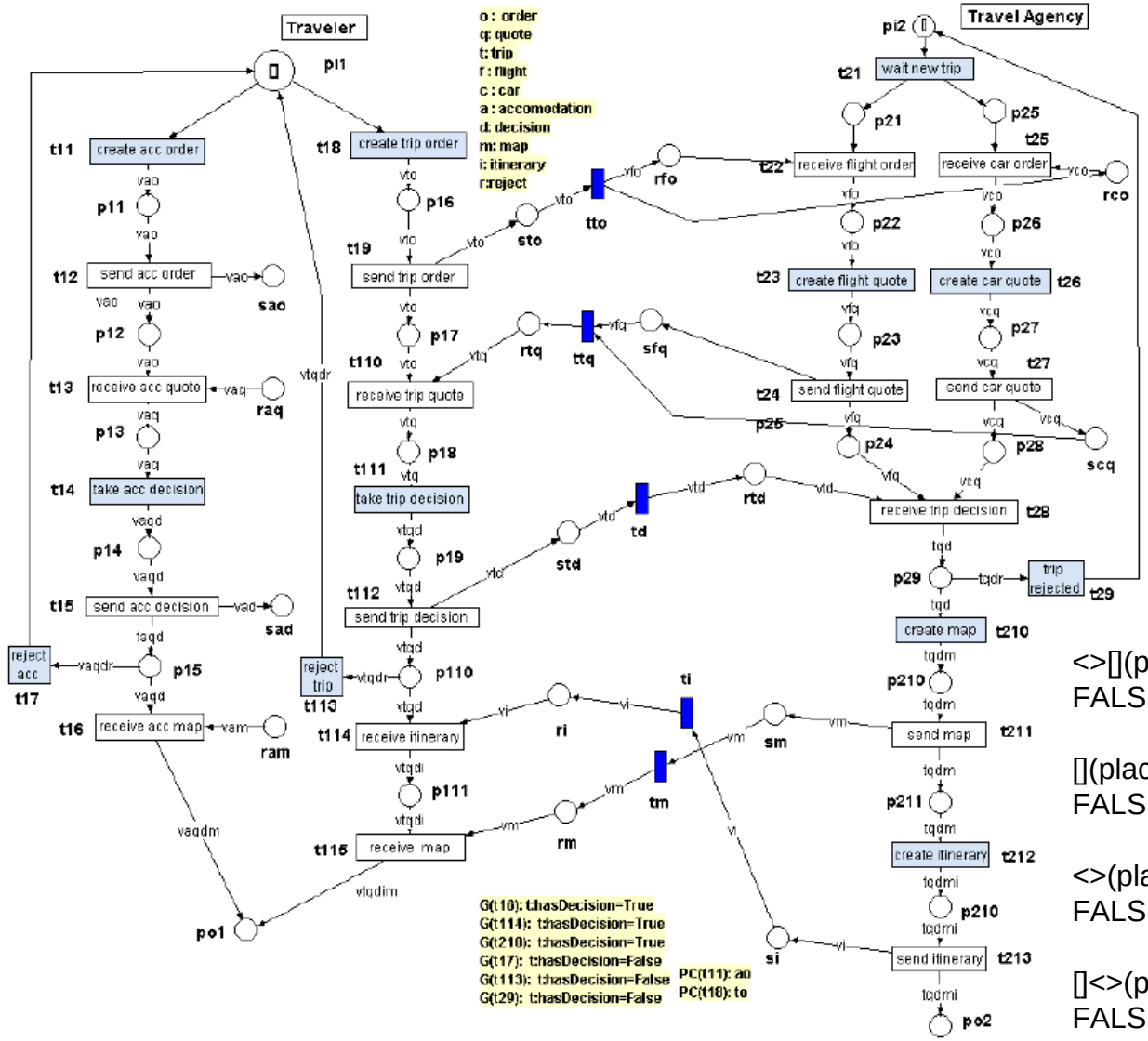



```
C:\maria>maria -b ExAdaptabilidadMed2.pn
deadlock state @8
deadlock state @44
```

```
@8:deadlock state (
 p12:
  1
 sao:
  1
 p21:
  1
 p25:
  1
)
2 predecessors
@0$show @44
@44:deadlock state (
 po1:
  1
 po2:
  1
)
1 predecessor
```

4. HERRAMIENTAS DE MODEL CHECKING

MEDIACIÓN DE PROCESOS WEB



<>[](place po1 equals 1) and (place po2 equals 1) FALSE

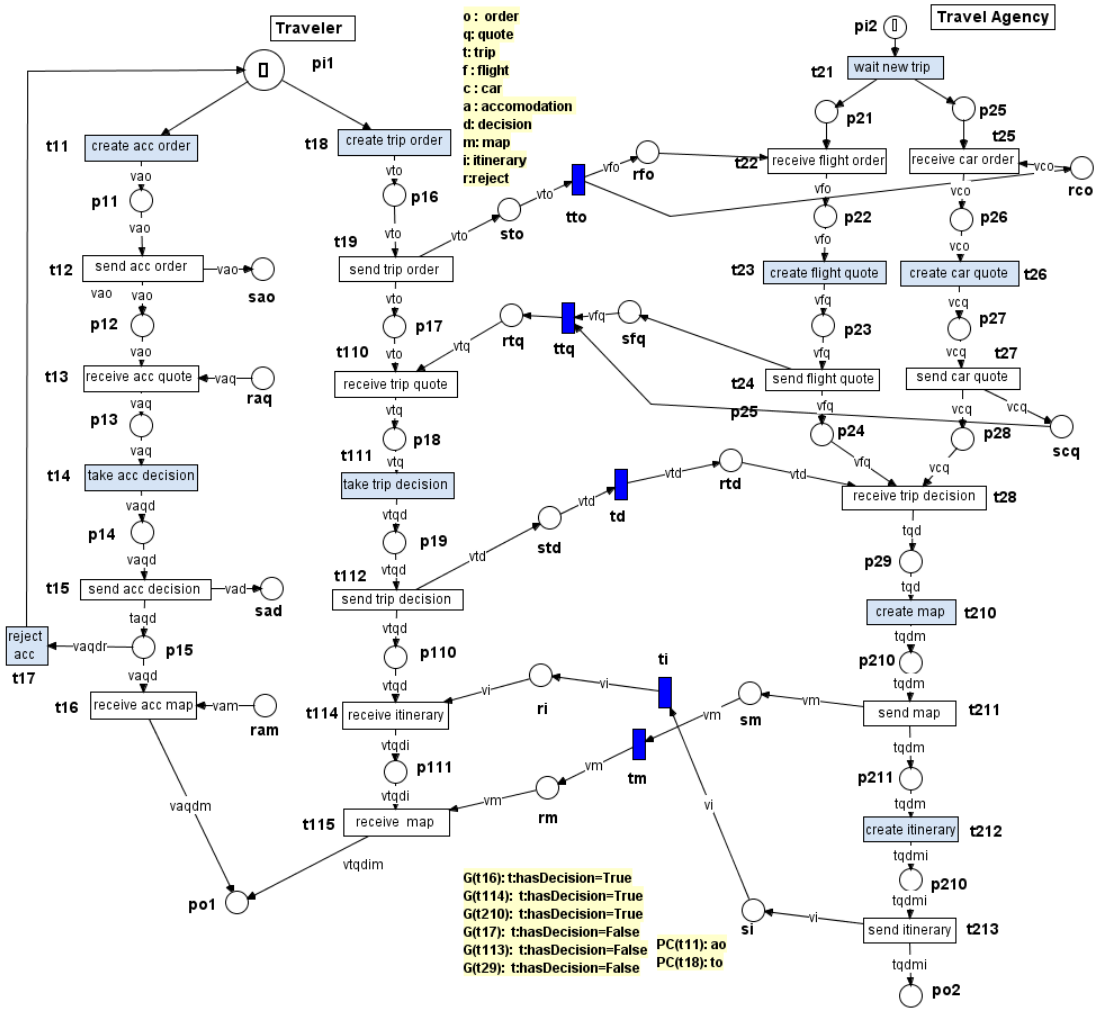
[](place po1 equals 1) and (place po2 equals 1) FALSE

<>(place po1 equals 1) and (place po2 equals 1) FALSE

[]<>(place po1 equals 1) and (place po2 equals 1) FALSE

4. HERRAMIENTAS DE MODEL CHECKING

MEDIACIÓN DE PROCESOS WEB



<>[](place po1 equals 1) and (place po2 equals 1)
TRUE

[](place po1 equals 1) and (place po2 equals 1)
FALSE

<>(place po1 equals 1) and (place po2 equals 1)
FALSE

[]<>(place po1 equals 1) and (place po2 equals 1)
FALSE

4. HERRAMIENTAS DE MODEL CHECKING

COMPATIBILIDAD DE PROCESOS WEB

Los Algoritmos de Model Checking son Lineales con respecto al numero de estados del modelo y con respecto al numero de operadores de la fórmula…

PERO el numero de estados puede ser exponencial con respecto al numero de elementos del modelo.
PROBLEMA DE EXPLOXIÓN DE ESTADOS