Departamento de Computación

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

TESIS DE LICENCIATURA

"Definición de un lenguaje de especificación para simulación de tráfico urbano siguiendo el paradigma Cell-DEVS"

Autora: Alejandra Davidson (L.U. 31/94)
Director: Dr. Gabriel Wainer

20 de mayo de 1999

Pabellón 1 - Planta Baja - Ciudad Universitaria
(1428) Buenos Aires
Argentina

http://www.dc.uba.ar

Índice

Resumen 4

Introducción 5

CAPÍTULO I - Simulación de Sistemas de Eventos Discretos usando Autómatas Celulares .. 128
1. Autómatas Celulares 129
2. Diversas aplicaciones modeladas con Autómatas Celulares 135
  2.1. Modelado del comportamiento de hormigas 135
  2.2. Asignación dinámica de canales en Telefonía Celular 139
  2.3. Wa-Tor 141
  2.4. Modelado de una cuenca fluvial 144
  2.5. Modelado de movimiento de peatones 147
3. Modelos de control de tráfico utilizando Autómatas Celulares 149
  3.1. Definición de una Taxonomía 150
  3.2. Clasificación de modelos según la taxonomía 152
  3.3. Discusión 199
4. Formalismos de Especificación 201
  4.1. DEVS 201
  4.2. Cell-DEVS 203

CAPÍTULO II - Definición de un lenguaje de especificación para simulación de tráfico urbano siguiendo el paradigma Cell-DEVS ..... **Error! Bookmark not defined.**
1. Lenguaje de Especificación y mapeo con Cell-DEVS ..... **Error! Bookmark not defined.**
  1.1. Estructura de las calles (Plano) ..... **Error! Bookmark not defined.**
    1.1.1. Calles ..... **Error! Bookmark not defined.**
      1.1.1.1. Calles de carril único ..... **Error! Bookmark not defined.**
      1.1.1.2. Calles de dos carriles de igual dirección ..... **Error! Bookmark not defined.**
      1.1.1.3. Calles de tres carriles de igual dirección ..... **Error! Bookmark not defined.**
      1.1.1.4. Calles de cuatro carriles de igual dirección ..... **Error! Bookmark not defined.**
      1.1.1.5. Calles con más de cuatro carriles de igual dirección **Error! Bookmark not defined.**
      1.1.1.6. Calles con carriles de distinta dirección ..... **Error! Bookmark not defined.**
    1.1.2. Cruces ..... **Error! Bookmark not defined.**
    1.1.3. Restricciones y generalidades del Lenguaje ..... **Error! Bookmark not defined.**
    1.1.4. Acoplamiento entre Cruces y Tramos ..... **Error! Bookmark not defined.**
  1.2. Elementos de control ..... **Error! Bookmark not defined.**
    1.2.1. Semáforos ..... **Error! Bookmark not defined.**
    1.2.2. Trenes ..... **Error! Bookmark not defined.**
      1.2.2.1. Definición de Trenes usando Cell-DEVS ..... **Error! Bookmark not defined.**
      1.2.2.2. Definición de Trenes usando DEVS individuales **Error! Bookmark not defined.**
    1.2.3. Obras ..... **Error! Bookmark not defined.**
    1.2.4. Baches ..... **Error! Bookmark not defined.**
    1.2.5. Señales de tránsito y otros elementos de control ..... **Error! Bookmark not defined.**
    1.2.6. Autos estacionados ..... **Error! Bookmark not defined.**
    1.2.7. Tasa de Choques ..... **Error! Bookmark not defined.**
    1.2.8. Marco Experimental ..... **Error! Bookmark not defined.**
2. Extensión del Lenguaje ..... **Error! Bookmark not defined.**
  2.1. Camiones ..... **Error! Bookmark not defined.**
    2.1.1. Calles con circulación de camiones ..... **Error! Bookmark not defined.**
    2.1.2. Cruces con circulación de camiones ..... **Error! Bookmark not defined.**
      2.1.2.1. Celdas de salida del cruce ..... **Error! Bookmark not defined.**
      2.1.2.2. Celdas de ingreso al cruce ..... **Error! Bookmark not defined.**

2.1.3. Acoplamiento entre Tramos y Cruces.....**Error! Bookmark not defined.**
2.2. Choques .....**Error! Bookmark not defined.**
3. Apéndices.....**Error! Bookmark not defined.**
3.1. Apéndice A.....**Error! Bookmark not defined.**
3.1.1. Funciones Auxiliares .....**Error! Bookmark not defined.**
3.1.2. Constantes y Funciones Aleatorias .....**Error! Bookmark not defined.**
3.2. Apéndice B - Descripción de las reglas de especificación **Error! Bookmark not defined.**
3.3. Apéndice C - Resumen del Lenguaje de Especificación de secciones de ciudades ..**Error! Bookmark not defined.**
3.4. Apéndice D - Lenguaje de Especificación para los Modelos**Error! Bookmark not defined.**
3.5. Apéndice E - Reglas y Vecindario de los tramos con camiones**Error! Bookmark not defined.**

Conclusiones .....**Error! Bookmark not defined.**

Bibliografía .....**Error! Bookmark not defined.**

Resumen

En los últimos años el formalismo de autómatas celulares ha sido utilizado para simulación de tráfico de vehículos. En este trabajo se analizan algunos de estos modelos, que se caracterizan por representar sólo aspectos básicos del problema, sin considerar la presencia de baches, barreras de trenes, señales de tránsito, autos estacionados, etc. Para facilitar su análisis, se propone una taxonomía que los clasifica según la estructura de las calles (cantidad de carriles, cruces, etc.) y según las características modeladas (semáforos, choques, etc.). Además se define un lenguaje de alto nivel para la especificación de secciones de ciudad que permite representar una gran variedad de características del tráfico urbano, cubriendo aspectos no considerados en los modelos analizados. Mediante este lenguaje se logra separar los modelos de simulación de la especificación del problema, permitiendo plantear secciones de ciudad de características diversas que se resuelven a través de un mapeo genérico definido para tal fin. De esta forma, se definen modelos DEVS y Cell_DEVS que representan cada una de las construcciones del lenguaje, manteniendo los lineamientos básicos que fueron analizados en los trabajos de Autómatas Celulares. El paradigma Cell_DEVS mejora las restricciones de precisión y performance impuestas por los AC. Por otro lado, también facilita la definición de reglas de comportamiento, que en general son más simples, permitiendo la incorporación de más aspectos del problema al modelo.

Introducción

Una *simulación* es la reproducción del comportamiento dinámico de un sistema real en base a un modelo, con el fin de llegar a conclusiones aplicables al mundo real. Su utilización se debe a que en muchos casos no se puede experimentar directamente sobre el sistema a estudiar, o se desea evitar costos, peligro, etc. Además permite experimentación controlada, compresión de tiempo (una simulación se realiza en mucho menos tiempo que el sistema real que modela), y análisis de sensibilidad. Otra gran ventaja es que su uso no afecta al sistema real, que puede seguir utilizándose (o no existir). Finalmente, la simulación es una herramienta efectiva de entrenamiento.

DEVS es un formalismo para modelar sistemas dinámicos complejos, de *tiempo continuo* (el tiempo se representa con una variable de tipo real) usando modelado de *eventos discretos* (las variables descriptivas del modelo toman sus valores dentro de un conjunto discreto). Para especificar modelos DEVS es conveniente ver al modelo como ports de entrada/salida que interactuan con el entorno. Cuando se reciben en los ports de entrada los eventos externos, la descripción del modelo debe determinar cómo responder. Un modelo DEVS se construye en base a un conjunto de modelos básicos (atómicos), que se combinan para formar modelos acoplados. Los *modelos atómicos* son objetos independientes modulares, con variables de estado y parámetros, funciones de transición internas, externas, de salida y avance de tiempo. Un *modelo acoplado* especifica cómo se conectan las entradas y salidas de los componentes. Los nuevos modelos también son modelos modulares, y pueden usarse para armar los modelos de mayor nivel.

Un *autómata celular* es un conjunto infinito n-dimensional de celdas ubicadas geométricamente, donde cada punto de la grilla (*celda*) puede tener un estado elegido de un alfabeto finito y evoluciona en el tiempo de acuerdo a un conjunto de *reglas locales* de transición bien definidas. Además todas las celdas contienen el mismo aparato de cálculo del nuevo estado y se conectan entre sí de forma uniforme. Es necesario definir la *vecindad* de una celda, que es un conjunto de celdas cercanas. Esta vecindad en general es homogénea para todas las celdas y se utiliza como parámetro para obtener el nuevo estado de las mismas.

Cell_DEVS es un paradigma de especificación utilizado para describir modelos celulares en forma de modelos atómicos DEVS con distintos tipos de demoras. Es decir, cada celda será definida como un modelo atómico DEVS que podrá tener asociada algún tipo de demora (inercial o de transporte). Luego estas celdas serán acopladas para formar un espacio Cell_DEVS completo a través de la relación de vecindad. Estos espacios a su vez pueden ser acoplados con otros espacios Cell_DEVS con distinta definición de la vecindad, con distinto comportamiento, o bien, con otros modelos de la jerarquía DEVS.

Un área de aplicación del formalismo de los Autómatas Celulares es la simulación de sistemas de control de tráfico de vehículos. La importancia de estos modelos es permitir, a través de su simulación, definir nuevas normas de tránsito, testear los efectos de la introducción de controles en distintas secciones, medir las consecuencias sobre el flujo del vehículo que provoca un choque o parte de una calle con hombres trabajando, controlar la polución ambiental provocada por los vehículos, evitar la generación de embotellamientos; entre otras cosas.

Para profundizar y avanzar sobre la especificación de estos sistemas se analizan los modelos existentes para la simulación del tráfico de vehículos que utilizan el formalismo de Autómatas Celulares. Estos modelos se caracterizan por comenzar especificando los aspectos básicos de la circulación de vehículos, y luego son extendidos a través del agregado de reglas que los acercan cada vez más a la realidad.

Se propone una taxonomía para facilitar el análisis de los modelos de control de tráfico estudiados y a través de ella se puede establecer qué características del sistema real fueron especificadas. Además en ella se consideran los aspectos principales que influencian el comportamiento de los vehículos y que deberían ser modelados para obtener resultados semejantes a la realidad.

Además se define un lenguaje de alto nivel para la especificación de secciones de ciudad que permite representar una gran variedad de características del tráfico urbano, cubriendo aspectos tales como circulación de autos y camiones, presencia de semáforos, choques, baches, etc. Es decir, se propone una aproximación más completa que la mayoría de los modelos existentes para la simulación de tráfico.

El lenguaje provee un nivel de abstracción que separa los modelos de simulación de la especificación del problema, permitiendo plantear secciones de ciudad de características diversas que se resuelven a través de un mapeo genérico definido para tal fin. Por lo tanto, el usuario del lenguaje no necesita interactuar con los modelos de simulación y por ende no es necesario que conozca los formalismos utilizados para especificarlos.

El lenguaje definido se mapea sobre los formalismos de especificación de modelos para simulación Cell-DEVS y DEVS. Para ello se proponen modelos que representan cada una de las construcciones del lenguaje, manteniendo los lineamientos básicos que fueron planteados para Autómatas Celulares en los trabajos analizados. Pero a diferencia de ellos se agregan más características del tráfico de vehículos y se realiza la menor cantidad de simplificaciones posibles sobre los modelos planteados.

CAPÍTULO II - Definición de un lenguaje de especificación para simulación de tráfico urbano siguiendo el paradigma Cell-DEVS 8
1. Lenguaje de Especificación y mapeo con Cell-DEVS 9
1.1. Estructura de Calles (Plano) 9
1.1.1. Calles 10
1.1.1.1. Calles de carril único 11
1.1.1.2. Calles de dos carriles de igual dirección 16
1.1.1.3. Calles de tres carriles de igual dirección 23
1.1.1.4. Calles de cuatro carriles de igual dirección 31
1.1.1.5. Calles con más de cuatro carriles de igual dirección 37
1.1.1.6. Calles con carriles de distinta dirección 40
1.1.2. Cruces 41
Celdas de salida del cruce 44
Celdas de ingreso al cruce 45
1.1.3. Restricciones y generalidades del Lenguaje 46
1.1.4. Acoplamiento entre Cruces y Tramos 47
1.2. Elementos de Control 49
1.2.1. Semáforos 49
1.2.2. Trenes 55
1.2.2.1. Definición de Trenes usando Cell-DEVS 55
1.2.2.2. Definición de Trenes usando modelos atómicos DEVS 63
1.2.3. Obras 65
1.2.4. Baches 71
1.2.5. Señales de tránsito y otros elementos de control 72
1.2.6. Autos estacionados 73
1.2.7. Tasa de Choques 77
1.2.8. Marco Experimental 78
2. Extensión del Lenguaje 82
2.1. Camiones 82
2.1.1. Calles con circulación de camiones 82
2.1.2. Cruces con circulación de camiones 93
2.1.2.1. Celdas de salida del cruce 96
2.1.2.2. Celdas de ingreso al cruce 97
2.1.3. Acoplamiento entre Tramos y Cruces 98
2.2. Choques 99
3. Apéndices 103
3.1. Apéndice A 103
3.1.1. Funciones Auxiliares 103
3.1.2. Constantes y Funciones Aleatorias 106
3.2. Apéndice B - Descripción de las reglas de especificación de los tramos 107
3.3. Apéndice C - Resumen del Lenguaje de Especificación de secciones de ciudad 111
3.4. Apéndice D - Lenguaje de Especificación para los Modelos 112
3.5. Apéndice E – Reglas y Vecindario de los tramos con camiones 113
Calles de carril único (TC1) 113
Calles de 2 carriles (TC2) 113
Calles de 3 carriles (TC3) 116
Calles de 4 carriles (TC4) 118
Calles de 5 carriles (TC5) 120
Calles de más de 5 carriles (TC) 121
Conclusiones 123
Bibliografía 125

CAPÍTULO II - Definición de un lenguaje de especificación para simulación de tráfico urbano siguiendo el paradigma Cell-DEVS

Como fue analizado en la sección 3 del capítulo I, un área de aplicación de la simulación está constituida por los sistemas de control de tráfico de vehículos. La importancia de estos modelos es permitir la definición de nuevas normas de tránsito, testear los efectos de la introducción de controles en distintas secciones, medir las consecuencias sobre el flujo del vehículo que provoca un choque o parte de una calle con hombres trabajando, controlar la polución ambiental provocada por los vehículos, evitar la generación de embotellamientos, etc. Un modelo para simulación de tráfico debe permitir la definición de calles, avenidas, cruces entre ellas, semáforos, cruces de trenes, señales de tránsito y cualquier otro aspecto que afecte el flujo de vehículos. Al incluir mayor cantidad de características del tráfico en el modelo, se espera obtener un comportamiento simulado más próximo a la realidad y por ende resultados más valiosos.

Los autómatas celulares han demostrado ser una herramienta útil para la simulación del tráfico, puesto que logran generar dinámicas de tráfico razonables. En la sección 3.2 del capítulo I de este trabajo, se muestra un informe sobre modelos que han sido desarrollados usando este formalismo. Por lo general éstos sólo incluyen los aspectos básicos del flujo de vehículos, es decir, se limitan a modelar el movimiento de automotores hacia adelante, permitiendo en algunos casos giros en las esquinas o cambios de carril; pero sin considerar la presencia de choques, obras, señales de tránsito, etc. Por otro lado, el formalismo de autómatas celulares utiliza una base de tiempo discreta, que tiene restricciones en la precisión y eficiencia de los modelos simulados. En el caso particular de autómatas celulares complejos, se requiere mucho tiempo de cómputo para obtener un grado de precisión razonable.

El paradigma Cell-DEVS planteado en [WG98] permite la descripción de modelos celulares a través de su definición como modelos atómicos DEVS con distintos tipos de demoras. Este formalismo resulta atractivo para modelar el tráfico puesto que a través de las demoras se pueden representar una gran variedad de características del problema (la velocidad de los vehículos, baches en la calle, elevaciones transversales, bocacalles, etc.). Esto permite en forma sencilla generar modelos más completos que los planteados para autómatas celulares, manteniendo los mismos lineamientos básicos. Otra ventaja de este formalismo es que el tiempo progresa en forma continua, lo que permite obtener mayor precisión y evitar períodos de inactividad de la simulación; mejorando el uso de los recursos de la computadora.

El objetivo del presente trabajo es definir un lenguaje de alto nivel para especificar secciones de ciudades; estableciendo la ubicación de las calles, sentido de circulación de los vehículos, cantidad de carriles, semáforos, etc. Además este lenguaje será mapeado a los formalismos Cell_DEVS y DEVS para obtener modelos simulables para sistemas de control de tráfico, realizando únicamente una especificación de alto nivel. Se pretende obtener modelos que permitan representar una gran variedad de las características del tráfico urbano sin realizar simplificaciones que los alejen demasiado del sistema real, para luego poder analizar su costo de diseño y ejecución en comparación con modelos simplificados. Estos últimos son los que tienen mayor difusión en la actualidad y se caracterizan por utilizar pocas reglas para modelar un comportamiento homogéneo en todo el espacio. Sería interesante analizar las dinámicas de flujos generados por ambos enfoques de modelado, evaluando el costo en eficiencia y tiempo de diseño que se paga para obtener una mayor precisión del modelo.

En las siguientes secciones se define este lenguaje de especificación de secciones de ciudad, y su mapeo sobre modelos Cell-DEVS. Se comienza con la definición de la estructura de calles y cruces en la sección 1.3, y luego se define el modelado de semáforos, trenes, obras, baches, algunas señales de tránsito y autos estacionados (sección 1.4). Todas estas características del tráfico urbano se mapean con modelos de estado binario. Luego, en la sección 2 se extiende el lenguaje para permitir el modelado de la circulación de camiones y representar los efectos de choques de vehículos; utilizando modelos Cell-DEVS de estado discreto natural.

1. Lenguaje de Especificación y mapeo con Cell-DEVS

En esta sección se define un lenguaje de especificación de alto nivel que permite describir la estructura de calles y vías de circulación de vehículos de una sección de ciudad, incluyendo características que determinan e influencian el tránsito normal. Éstas últimas modelan la presencia de semáforos, baches, obras, cruces de trenes, etc. El lenguaje se define como una serie de conjuntos que representan todos estos aspectos.

Al presentar cada construcción del lenguaje, se especificarán los modelos DEVS o Cell-DEVS utilizados para representarlas. Este mapeo permite asegurar correctitud de los simuladores desarrollados, permitiendo que el modelador se concentre en la especificación del modelo y la resolución de problemas concretos, abstrayéndose de las simulaciones.

Una sección de ciudad especificada con este lenguaje, será representada con un modelo en el que los vehículos avanzan (derecho o en diagonal) sobre alguna calle hasta llegar a las esquinas. Aquí deberán ingresar al cruce, verificando que haya espacio suficiente en el mismo.

Los cruces se representan como un anillo de celdas donde los vehículos giran en sentido contrario a las agujas del reloj hasta que deciden salir (ingresando en otra calle), como fuera planteado en [CQL95], [CLQ96] y [CDL97], y resumido en la sección 3.2 del capítulo I. La salida se modela con una función aleatoria local a la celda. Los vehículos que se encuentran dentro del cruce tienen prioridad de movimiento sobre los que desean ingresar. La velocidad se representa a través de una demora de transporte y se utiliza una función aleatoria para determinarla.

Luego se puede especificar la presencia de semáforos, barreras de trenes, obras y otros elementos de control que modificarán el comportamiento básico del modelo.

Cabe destacar que todos los modelos planteados en esta sección para representar cada construcción del lenguaje se definen utilizando Cell-DEVS de estado binario. Esto facilita la validación de las reglas de comportamiento que se puede lograr considerándolas como expresiones booleanas y utilizando operadores binarios únicamente (ejecución eficiente). Se podrían encontrar sin demasiado esfuerzo, inconsistencias en tiempo de compilación, tales como 2 reglas distintas que se puedan aplicar y que llevan a diferentes estados. Además, la cantidad de memoria necesaria para almacenamiento es pequeña. Pero la gran desventaja frente a los modelos no binarios es que éstos últimos tienen mayor poder expresivo, que permite modelar una gran variedad de situaciones.

1.3. Estructura de Calles (Plano)

En esta sección se brindan las construcciones del lenguaje de especificación que permiten definir el plano de la sección de ciudad que se quiere modelar. Para ello se debe establecer la ubicación y longitud de las calles, sus intersecciones o cruces, la dirección de circulación de los vehículos, etc. Se comienza con la descripción de las calles en la sección 1.3.1 y luego los cruces en 1.3.2; siempre definiendo el modelo Cell-DEVS correspondiente a cada la construcción del lenguaje que se introduce.

1.3.1. Calles

Una calle se especifica como una secuencia de *tramos*, cada uno de éstos representa un sector que se extiende a lo largo de una cuadra, donde todos los carriles tienen la misma dirección de circulación y una velocidad máxima permitida. Es decir, constituyen una sección de la calle sin cruces y de mano única. Consecuentemente para construir un sector *doble mano* es necesario definir un tramo en cada dirección.

Los tramos se pueden especificar como:

$$Tramos = \{(p1, p2, n, a, dir, max) / p1, p2 \in Puntos \wedge p1 \neq p2 \wedge n, max \in N \wedge a, dir \in \{0, 1\}\}$$

Por lo tanto, para cada tramo o elemento de este conjunto se deben identificar sus extremos, la cantidad de carriles, si tendrá forma recta o curva, el sentido de circulación de los vehículos y finalmente, su velocidad máxima permitida. Entonces, un tramo t es una tupla de seis elementos:

$$t = (p1, p2, n, a, dir, max)$$

donde,

- p1 y p2, representan los extremos del tramo y pertenecen al conjunto Puntos que se define como:

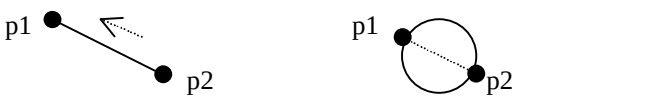
  $$Puntos = \{ (x,y) / x, y \in Z \}$$

  Se pide $p1 \neq p2$, pues en el modelo no tiene sentido un tramo de longitud 0;
- $n \in N$, indica la cantidad de carriles del tramo;
- $dir \in \{0,1\}$, indica el sentido de circulación de los vehículos. Si dir = 1 los vehículos se desplazan hacia p2, caso contrario lo hacen hacia p1;
- $a \in \{0, 1\}$, indica la forma del tramo. Aquí:

  $a = 0 \Rightarrow$ el tramo es un segmento recto determinado por los puntos p1 y p2.

  $a = 1 \Rightarrow$ el tramo queda definido por una de las semicircunferencias que se obtienen al partir la circunferencia cuyo diámetro es el segmento que une a p1 y p2. La circunferencia se parte utilizando el segmento que une a p1 y p2 como eje del corte, obteniendo 2 semicircunferencias con modelo subyacente idéntico y por ende no es necesario que la especificación diferencie ambos casos (este detalle quedará en el nivel de la interfaz con el modelador).
- $max \in N$, indica la velocidad máxima de circulación permitida en el tramo.

En la siguiente figura se muestra un tramo con forma curva y otro con forma recta, según la definición anterior.



t = (p1, p2, n, 0, 0, max) t = (p1, p2, n, 1, dir, max)

Figura 1 – Especificación de Tramos

El movimiento de los vehículos sobre un tramo consiste en avanzar hacia adelante, ya sea recto o en diagonal. Esto no es posible si hay 1 ó 2 carriles, puesto que en el primer caso el auto sólo se puede mover en línea recta, y en el otro puede avanzar en línea recta o en una sola diagonal. Además los carriles de los bordes tampoco presentan el comportamiento completo y se deben diferenciar distintos modelos. Por esto, es necesario definir un modelo para tramos de un sólo carril, de 2 carriles y así hasta obtener el genérico para más de 4 carriles. En todos los casos se mantienen los lineamientos básicos del movimiento de los vehículos, adaptándolo a las dimensiones del espacio.

Las reglas del movimiento de los vehículos establecen que un auto siempre intenta moverse hacia adelante, si no hay suficiente espacio para esa maniobra intenta hacia adelante en diagonal izquierda y si ninguno de los 2 movimientos anteriores son posibles, intenta con la diagonal derecha. Por último, si no ha logrado avanzar por la falta de espacio entonces conservará su posición. Luego, los vehículos que avanzan derecho (sobre su propio carril) tienen prioridad de hacerlo frente a algún otro que quiera acceder a la misma posición desde otro carril. Además los automóviles que avanzan en diagonal izquierda tienen prioridad sobre los que quieren acceder a la misma posición avanzando en diagonal derecha. Por lo tanto, para que un vehículo se pueda mover derecho sólo debe pedir que haya lugar adelante. Para moverse al carril izquierdo debe verificar que tenga lugar (celda vacía) y que no haya otro auto que quiera acceder a la misma posición con movimiento recto. Por último, para un vehículo se pueda mover al carril derecho debe verificar que tenga lugar (celda vacía) y que no haya otro auto que quiera acceder a la misma posición con movimiento recto o en diagonal izquierda.

En la siguientes secciones se definen los tramos asumiendo que se encuentran acoplados a cruces en los 2 extremos. Más adelante, al describir el marco experimental (sección 1.4.8), se definen los modelos para los tramos que no tienen acoplado un cruce en alguno sus extremos y que representan las entradas y salidas de los vehículos en el modelo a simular.

1.3.1.1. Calles de carril único

Un tramo t = (p1, p2, 1, a, dir, max) de carril único se define como un modelo Cell_DEVS de una dimensión con demora de transporte, cuya estructura se presenta en la Figura 2.

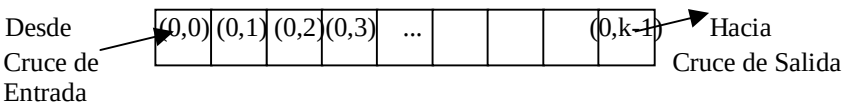


Figura 2 – Tramo de carril único

Cada celda en este espacio se define como:

$$C_{0j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = $<\eta, P^x, P^y>$, donde

η = 3; y P^x y P^y se definen como en la sección 4 del capítulo I; es decir,
P^x = { (X_1, binario), (X_2, binario), (X_3, binario) };
P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario) }.

$\boldsymbol{X} = \boldsymbol{Y}$ = {0, 1};
S:

s = { 1 si hay un vehículo en la celda;
    { 0 sino.

N = { (0,-1), (0,0), (0,1) };

(0,-1)	(0,0)	(0,1)

Figura 3 - Vecindario de la celda origen

delay = transport;

d = Conversión_Demora(velocidad(max));
donde,

- velocidad: N → N, es una función aleatoria con la distribución probabilística característica del flujo de vehículos. Se espera que pocos autos circulen con las velocidades extremas (máxima y mínima) y que la mayoría lo haga con velocidades intermedias. Recibe como parámetro el valor máximo que puede alcanzar la función y en base a él se obtiene la media $\bar{x} = \frac{2}{3} * max(km/h)$ y desvío estándar $\sigma = \frac{1}{3} * max(km/h)$, en este caso max es el parámetro pasado a la función. El valor que retorna es un natural que representa una velocidad en km/h elegida en forma aleatoria de acuerdo a la distribución descripta.
- max es la constante especificada en t, que representa la velocidad máxima de circulación permitida, en km/h.
- Conversión_Demora es una función detallada en el Apéndice A, que recibe como parámetro un valor natural que representa una velocidad (en km/h) y devuelve el tiempo (en segundos) que se debe demorar el auto en la celda para representar que avanza con esa velocidad.

λ, δ_{int} y δ_{ext} se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte, (ver sección 4.2 del capítulo I).

Notación 1:

- Para la descripción de los modelos se utiliza el lenguaje presentado en el Apéndice D para espacios Cell-DEVS de dos dimensiones.
- Cada regla que modela el comportamiento de los autos en los tramos, recibirá un nombre con el que será referenciada a lo largo del trabajo (en el Apéndice B se resumen las reglas con su nombre, especificación y explicación).

τ: S x N → S se define de la siguiente manera,

Nuevo estado	Estado del vecindario	Nombre de la regla
1	(0,-1) = 1 and (0,0) = 0	Llega_Desde_Atrás

0	(0,0) = 1 and (0,1) = 0	Sale_Hacia_Adelante
(0,0)	t /*en otro caso conserva estado*/	Default

Analicemos con detalle esta especificación. El estado de una celda representa la presencia (s = 1) o ausencia (s = 0) de un vehículo. En un tramo de carril único sólo se permite que los vehículos avancen hacia la posición de adelante, siempre y cuando la misma esté vacía. De esta forma el vecindario necesario para establecer el nuevo estado de una celda está constituido por ella misma, la celda anterior y la siguiente.

Las demoras de transporte se utilizan para modelar las demoras provocadas por la aceleración de los automotores. El movimiento de los mismos se demora antes del siguiente movimiento a la próxima celda. Este valor es generado mediante una función aleatoria que permite modelar distintas velocidades para cada vehículo en distintos momentos.

El movimiento de los vehículos (especificado con la función τ) queda definido por 3 reglas. La regla Llega_Desde_Atrás representa el movimiento recto del vehículo desde la celda de atrás hacia la origen, siempre y cuando ésta se encuentre vacía. La regla Sale_Hacia_Adelante representa el movimiento recto del vehículo que se encuentra en la celda origen hacia la de adelante. La regla Default considera cualquier otro caso donde el estado de la celda no cambia.

El modelo acoplado correspondiente al tramo t = (p1, p2, 1, a, dir, max) se define como:

$$TC1(k, max) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z, select >$$

donde,

Ylist = { (0,0), (0,k-1) }

Xlist = { (0,0), (0,k-1) }

I = $<P^x, P^y>$

$P^x = \{<X_{\eta+1}(0,0), binario>, <X_{\eta+1}(0,k-1), binario>\}$
$P^y = \{<Y_{\eta+1}(0,0), binario>, <Y_{\eta+1}(0,k-1), binario>\}$

De aquí en más, estos ports serán denotados con la siguiente convención:

Convención 1:

El nombre de los ports externos para los modelos se denota como t-n-c, donde:

- t indica si es un port de entrada (t = x) o si es un port de salida (t = y).
- n es la inicial (o las 2 primeras letras) del nombre del modelo de origen si es port de entrada (X) o de destino si es port de salida (Y). Por ejemplo, n = c representa a los cruces, n = t a los tramos, n = se a los semáforos, etc.
- c es una cadena de caracteres que representa alguna característica particular del port.

Por ende, los ports se denominarán de la siguiente forma:

Port	Nombre	Comentario
$X_{\eta+1}(0,0)$	x-c-hayauto	Este port se utiliza para informar de la salida de un auto desde el cruce hacia el

		tramo (portvalue($X_{\eta+1}$) = 1).
$X_{\eta+1}$(0,k-1)	x-c-haylugar	Este port permite saber si en el cruce hay lugar (portvalue($X_{\eta+1}$) = 0) para que avance un auto desde el tramo.
$Y_{\eta+1}$(0,0)	y-c-haylugar	Este port se utiliza para que el cruce pueda saber si hay lugar en el tramo para que un auto pueda salir de él (portvalue($Y_{\eta+1}$) = 0).
$Y_{\eta+1}$(0,k-1)	y-c-hayauto	Este port informa al cruce de la presencia de un auto en el tramo que desea avanzar hacia él (portvalue($Y_{\eta+1}$) = 1).

$\boldsymbol{X}$ = { 0, 1 }

$\boldsymbol{Y}$ = { 0, 1 }

n = 1

$\mathbf{t_1}$ = k

$\boldsymbol{\eta}$ = 3

N = { (0,-1), (0,0), (0,1) }

C = { C_{ij} / i = 0 $\wedge$ j $\in$ [0, k-1] }, donde cada C_{ij} es un componente Cell-DEVS con demora como los definidos en la sección 4.2 del capítulo I. La especificación de cada celda ha sido descripta antes de introducir el modelo acoplado.

B = { (0,0), (0,k-1) }

Z se construye siguiendo la definición dada por el formalismo Cell_DEVS, instanciada con el vecindario de este espacio, (ver sección 4.2 del capítulo I).

select = { (0,1), (0,0), (0,-1) }

TC1(k, max) significa Tramo de 1 Carril de longitud k (celdas) con velocidad máxima de circulación de max (Km/h). Ambos parámetros son necesarios para la construcción del modelo y se derivan de la especificación del tramo t, en particular la velocidad máxima está explícitamente en ella; mientras que la cantidad de celdas se obtiene como:

k = Ctd_Celdas(t)

Donde la función Ctd_Celdas se encuentra definida en el Apéndice A y establece este parámetro calculando la longitud del tramo (a partir de p1 y p2), y luego dividiéndola por el tamaño definido para la celda.

La especificación de este modelo representa el movimiento de los vehículos en calles de carril único. Cada tramo se extiende entre un cruce de origen y otro de destino, por lo tanto se definen los ports que acoplan estos modelos y permiten el avance de tráfico de uno al otro. La interfaz la conforman las celdas (0,0) y (0,k-1), pues cada una debe intercambiar información con los modelos de los cruces correspondientes.

El comportamiento para las celdas borde es distinto al definido anteriormente. Para la celda (0,0) se define el siguiente vecindario y comportamiento:

$\boldsymbol{\eta}$ = 2

N = { (0,0), (0,1) }

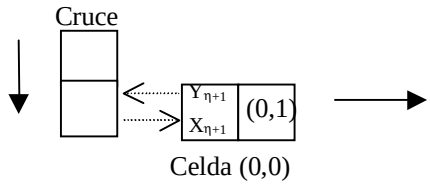


Figura 4- Vecindario y acoplamiento de la celda (0,0)

Notación 2:

Para la descripción de los modelos se utilizan 2 funciones relacionadas la interfaz del espacio, ellas son: *portvalue* y *send*. La primera recibe como parámetro el nombre de un port externo y retorna su valor actual. Mientras que la segunda, permite establecer el nuevo estado de los ports externos seleccionados. Es decir, send (v, P_{j1}^{y} , P_{j2}^{y} , ... , P_{jn}^{y}) representa que el nuevo estado de los ports {P_{j1}^{y} , P_{j2}^{y} , ... , P_{jn}^{y}} es v, donde v $\in$ {0, 1}.

Convención 2:

Cuando no se define send en forma explícita se asume que *todos* los ports de salida son actualizados con el nuevo valor del estado de la celda.

Convención 3:

En las figuras donde se grafican los modelos, las flechas rellenas representan el sentido de circulación de los vehículos mientras que las flechas punteadas muestran la interconexión de los Ports.

La función τ para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nombre de la regla
0	((0,0) = 1 and (0,1) = 0)	Sale_Hacia_Adelante
1	(portvalue(x-c-hayauto) = 1 and (0,0) = 0)	Llega_Desde_Cruce
(0,0)	t /*en otro caso conserva estado*/	Default

Los demás parámetros del modelo para esta celda no cambian.

La celda (0,0) tiene el vecindario formado por ella y la celda de adelante, además se encuentra acoplada a la celda del cruce desde la que recibe los vehículos. Esto se puede ver en la Figura 4. Para conocer el estado de la celda del cruce se utiliza el port x-c-hayauto ($X_{\eta+1}$), siempre que esté en 1 representa que existe un vehículo que saldrá del cruce hacia la celda (0,0). Por lo tanto, la regla Llega_Desde_Cruce representa el movimiento de avance de un vehículo que se encuentra en el cruce (portvalue(x-c-hayauto) = 1) hacia la celda (0,0). Las otras 2 reglas son las mismas que fueron definidas para el resto de las celdas. La celda (0,0) envía su estado hacia el cruce a través del port y-c-haylugar, es decir 0 si está vacía y 1 en otro caso.

Para la celda (0,k-1) también se define un vecindario y comportamiento diferente al resto. Aquí:

η = 2

N = { (0,-1), (0,0) }

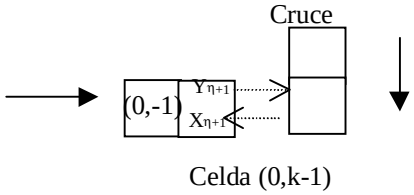


Figura 5- Vecindario y acoplamiento de la celda (0,k-1)

La función τ y el tipo de demora (delay) para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports	Delay	Nombre de la regla
0	((0,0) = 1 and portvalue(x-c-haylugar) = 0)	send(1, y-c-hayauto)	inercial	Sale_Hacia_Cruce
1	((0,-1) = 1 and (0,0) = 0)	send(0, y-c-hayauto)	transport	Llega_Desde_Atrás
(0,0)	t /*en otro caso conserva estado */	send(0, y-c-hayauto)	transport	Default

Los demás parámetros del modelo para esta celda no cambian.

La celda (0,k-1) tiene el vecindario formado por ella y la celda de atrás, además se encuentra acoplada con 1 celda del cruce. Esto se puede ver en la Figura 5. El valor que recibe a través del port externo es 1 si no hay suficiente espacio para el ingreso de un auto al cruce, esto es que la celda de ingreso o la de atrás están ocupadas; en otro caso recibe un 0. Los vehículos desde la celda (0,k-1) avanzan hacia el cruce. Según se ha definido el modelo, para que un vehículo pueda ingresar al cruce deberá chequear que la celda a la que quiere ingresar y la anterior a ésta se encuentren vacías, pues si un vehículo dentro del cruce quiere acceder a la misma posición éste tendrá prioridad. De esta forma el port x-c-haylugar informa del estado de la celda de ingreso y de la anterior. Por lo tanto, la regla Sale_Hacia_Cruce representa el avance del vehículo desde la celda (0,k-1) hacia la de adelante (dentro del cruce), siempre y cuando ésta y su anterior se encuentren vacías (portvalue(x-c-haylugar) = 0). Las otras 2 reglas son las mismas que fueron definidas para las celdas con comportamiento normal. El valor que envía la celda (0,k-1) por el port de salida es 1 si existe un auto en la celda que está habilitado (las celdas del cruce permanecieron vacías durante la demora inercial correspondiente) para avanzar al cruce, caso contrario envía 0.

La celda (0,k-1) se modela con los 2 tipos de demora (inercial y transporte) para representar diferentes comportamientos. Por un lado, cuando un vehículo quiere avanzar hacia la celda (0,k-1), regla Llega_Desde_Atrás, se utiliza demora de transporte, pues este movimiento no varía del realizado en cualquier otra celda del tramo a pesar que (0,k-1) esté en el borde. Pero para ingresar al cruce, es decir abandonar la celda (0,k-1), se modela con demora inercial porque representa el comportamiento de los vehículos al llegar a las esquinas y en general este tipo de maniobra se realiza con más precaución que el avance sobre el resto de la calle. Los automotores avanzan sobre el cruce si durante un cierto tiempo (demora inercial) tienen el camino libre, caso contrario esperan a que pasen los vehículos que ya se encuentran en el mismo. De esta forma los autos que circulan dentro del cruce tienen mayor probabilidad de avanzar que aquellos que desean ingresar y estos últimos no lo hacen si no tienen el camino libre por el tiempo que les lleva realizar la maniobra.

1.3.1.2. Calles de dos carriles de igual dirección

Un tramo t = (p1, p2, 2, a, dir, max) se define como un modelo Cell_DEVS de dos dimensiones con demora de transporte, cuya estructura se presenta en la Figura 6.

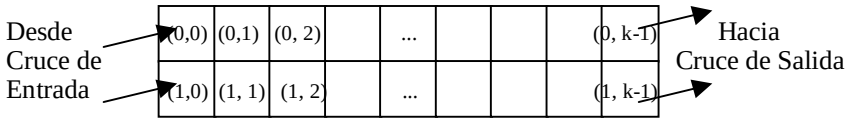


Figura 6 – Tramo de 2 carriles

Las celdas de cada fila de este espacio tienen comportamiento distinto, determinado por la conformación de su vecindario, y serán especificadas por separado. Es decir, los vehículos de las celdas de la primera fila sólo se pueden mover hacia adelante y en diagonal derecha; mientras que los de la otra lo pueden hacer hacia adelante y en diagonal izquierda.

Notación 3

~~Las celdas de la i-ésima fila del espacio serán llamadas celdas del carril i_ésimo.~~

Las celdas de la primera fila (carril 0) del espacio se definen como:

$$C_{0j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = < η, P^x, P^y>, donde

η = 6;

P^x = { (X_1, binario), (X_2, binario), (X_3, binario), (X_4, binario), (X_5, binario), (X_6, binario) };

P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario), (Y_4, binario), (Y_5, binario) , (Y_6, binario) }.

X = ***Y*** = {0, 1};

S:

$$s = \begin{cases} 1 \text{ si hay un vehículo en la celda;} \\ 0 \text{ sino.} \end{cases}$$

delay = transport;

d = Conversión_Demora(velocidad(max));

λ, **δ_{int}** y **δ_{ext}** se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

N = { (0,0), (0,1), (-1,0), (-1,1), (0,-1), (-1,-1) };

(0,-1)	(0,0)	(0,1)	carril 0
(-1,-1)	(-1,0)	(-1,1)	carril 1

Figura 7 - Vecindario de la celda origen (carril 0)

τ se define como:

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (-1,-1) = 1 and (-1,0) = 1)	Llega_Desde_CarrilDer
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (-1,1) = 0 and (-1,0) = 0)	Sale_Hacia_CarrilDer_ConPrioridad
(0,0)	t /*en otro caso conserva estado */	Default

Los autos que avanzan hacia estas celdas sólo lo pueden hacer desde la celda de atrás o desde el otro carril (derecho), debido a la conformación del vecindario. Para los movimientos en diagonal se debe considerar que un vehículo primero intenta moverse derecho y que tiene prioridad de acceso a las posiciones de su propio carril frente a los coches de otros carriles. Así, la regla Llega_Desde_CarrilDer representa que un vehículo avanza hacia la celda origen desde atrás en diagonal y pide que no haya nadie que pueda acceder al mismo lugar desde atrás derecho ((0,-1) = 0) y además que el auto que quiere ocuparla no pueda avanzar sobre su carril ((-1,-1) = 1 and (-1,0) = 1). Luego, los autos que abandonan estas celdas lo pueden hacer hacia adelante o hacia la diagonal derecha. La regla Sale_Hacia_CarrilDer_ConPrioridad representa que el vehículo de la celda origen avanza en diagonal siempre y cuando no haya un auto que avance derecho hacia la celda destino ((-1,0) = 0).

Las celdas de la segunda fila (carril 1) del espacio se definen como:

$$C_{1j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = < η, P^x, P^y>, donde

η = 6;

P^x = { (X_1, binario), (X_2, binario), (X_3, binario), (X_4, binario), (X_5, binario), (X_6, binario) };

P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario), (Y_4, binario), (Y_5, binario) , (Y_6, binario) }.

X = ***Y*** = {0, 1};

S:

s = { 1 si hay un vehículo en la celda;

⌊ 0 sino.

delay = transport;

d = Conversión_Demora(velocidad(max));

λ, δ_{int} y δ_{ext} se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

N = { (0,0), (0,1), (1,0), (1,1), (0,-1), (1,-1) };

(1,-1)	(1,0)	(1,1)	carril 0
(0,-1)	(0,0)	(0,1)	carril 1

Figura 8 - Vecindario de la celda origen (carril 1)

τ se define como:

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1)	Llega_Desde_CarrilIz_Con Prioridad
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (1,1) = 0 and (1,0) = 0)	Sale_Hacia_CarrilIz
(0,0)	t /*en otro caso conserva estado */	Default

Los autos que avanzan hacia estas celdas sólo lo pueden hacer desde la celda de atrás o desde el otro carril (izquierdo), debido a la conformación del vecindario. Para los movimientos en diagonal se debe considerar que un vehículo primero intenta moverse derecho y que tiene prioridad de acceso a las posiciones de su propio carril frente a los coches de otros carriles. Así, la regla Llega_Desde_CarrilIz_ConPrioridad representa que un vehículo avanza hacia la celda origen desde atrás en diagonal y pide que no haya nadie que pueda acceder al mismo lugar desde atrás derecho ((0,-1) = 0) y además que el auto que quiere ocuparla no pueda avanzar sobre su carril ((1,-1) = 1 and (1,0) = 1). Luego, los autos que abandonan estas celdas lo pueden hacer hacia adelante o hacia la diagonal izquierda. La regla Sale_Hacia_CarrilIz representa que el vehículo de la celda origen avanza en diagonal siempre y cuando no haya un auto que avance derecho hacia la celda destino ((1,0) = 0).

El modelo acoplado correspondiente al tramo t = (p1, p2, 2, a, dir, max) se define como:

$$TC2(k, max) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z, select >$$

donde,

Ylist = { (0,0), (1,0), (0,k-1), (1,k-1) }

Xlist = { (0,0), (1,0), (0,k-1), (1,k-1) }

I = <P^x, P^y>

P^x = {<$X_{\eta+1}(0,0)$, binario>, <$X_{\eta+1}(1,0)$, binario>, <$X_{\eta+1}(0,k-1)$, binario>, <$X_{\eta+1}(1,k-1)$, binario>}

P^y = {<$Y_{\eta+1}(0,0)$, binario>, <$Y_{\eta+1}(1,0)$, binario>, <$Y_{\eta+1}(0,k-1)$, binario>, <$Y_{\eta+1}(1,k-1)$, binario>}

Estos ports serán denotados de la siguiente forma:

Port	Nombre
$X_{\eta+1}(i,0)$, $0 \le i \le 1$	x-c-hayauto
$X_{\eta+1}(i,k-1)$, $0 \le i \le 1$	x-c-haylugar
$Y_{\eta+1}(i,0)$, $0 \le i \le 1$	y-c-haylugar
$Y_{\eta+1}(i,k-1)$, $0 \le i \le 1$	y-c-hayauto

X = { 0, 1 }

Y = { 0, 1 }

n = 2

$\mathbf{t_1}$ = 2

$\mathbf{t_2}$ = k

$\boldsymbol{\eta}$ = 6

N ha sido descripto al definir el modelo de celda de cada carril.

C = { C_{ij} / $i \in [0, 1] \wedge j \in [0, k-1]$ }, donde cada C_{ij} es un componente Cell-DEVS con demora y la especificación de cada celda ha sido descripta antes de introducir el modelo acoplado.

B = { (0, k-1), (1, k-1), (0,0), (1,0) }

Z se construye siguiendo la definición dada por el formalismo Cell_DEVS, instanciada con el vecindario de este espacio.

select = { (0,1), (1,1), (0,0), (1,0), (0,-1), (1,-1) }

TC2(k, max) significa Tramo de 2 Carriles de longitud k (celdas) cada uno, con velocidad máxima max (en Km/h). Donde k = Ctd_Celdas(t) y max es la constante especificada en t.

El modelo para calles de 2 carriles es una extensión del planteado para tramos de carril único, permitiendo que los vehículos además de moverse a la celda de adelante, lo puedan hacer hacia la que tienen adelante en diagonal. Cada tramo se extiende entre un cruce de origen y otro de destino, por lo tanto se definen los ports que acoplan estos modelos y permiten el avance de tráfico de uno al otro. La interfaz de este modelo la conforman las celdas de la primera y última columna del espacio, pues cada una debe intercambiar información con los modelos de los cruces correspondientes.

El comportamiento para las celdas borde es distinto al definido anteriormente. Para la celda (0,0) se define el siguiente vecindario y comportamiento:

η = 4

N = { (0,0), (0,1), (-1,0), (-1,1) }

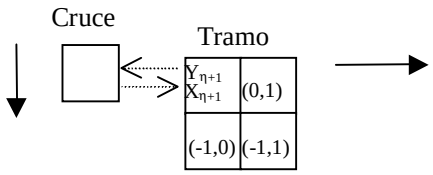


Figura 9 – Vecindario y acoplamiento de la celda (0,0)

La función τ para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and portvalue(x-c-hayauto) = 1)	Llega_Desde_Cruce
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (-1,1) = 0 and (-1,0) = 0)	Sale_Hacia_CarrilDer_Con Prioridad
(0,0)	t /*en otro caso conserva estado */	Default

Esta celda sólo recibe vehículos que salen del cruce, mientras que las reglas para avanzar desde ella son las mismas que las definidas para las celdas del carril 0. Los demás parámetros del modelo para esta celda no cambian.

Para la celda (1,0) también se define un vecindario y comportamiento diferente al resto. Aquí:

η = 4

N = { (0,0), (0,1), (1,0), (1,1) }

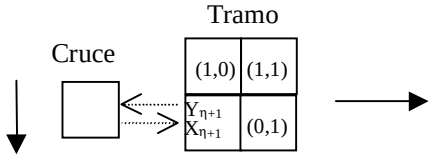


Figura 10 – Vecindario y acoplamiento de la celda (1,0)

La función τ para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and portvalue(x-c-hayauto) = 1)	Llega_Desde_Cruce
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (1,1) = 0 and (1,0) = 0)	Sale_Hacia_CarrilIIz
(0,0)	t /*en otro caso conserva estado */	Default

Esta celda sólo recibe vehículos que salen del cruce, mientras que las reglas para avanzar desde ella son las mismas que las definidas para las celdas del carril 1. Los demás parámetros del modelo para esta celda no cambian.

La utilización de los ports externos y la regla de avance de autos desde el cruce (Llega_Desde_Cruce) para las celdas de la columna 0 es la misma que se ha definido para la celda (0,0) del tramo de carril único (sección 1.3.1.1).

Para la celda (0,k-1) se define el siguiente vecindario y comportamiento:

$\eta = 4$

N = { (-1,0), (-1,-1), (0,0), (0,-1) }

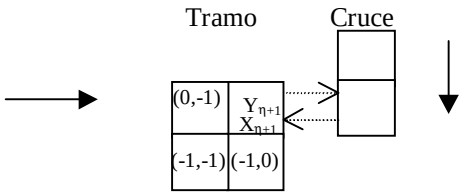


Figura 11 – Vecindario y acoplamiento de la celda (0,k-1)

La función τ y el tipo de demora (delay) para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports	Delay	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	send(0, y-c-hayauto)	transport	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (-1,0) = 1 and (-1,-1) = 1)	send(0, y-c-hayauto)	transport	Llega_Desde_CarrilDer
0	((0,0) = 1 and portvalue(x-c-haylugar) = 0)	send(1, y-c-hayauto)	inercial	Sale_Hacia_Cruce
(0,0)	t /*en otro caso conserva estado */	send(0, y-c-hayauto)	transport	Default

Las reglas para avanzar hacia esta celda son las mismas que las definidas para las celdas del carril 0; mientras que para abandonar la celda, se debe ingresar al cruce. Los demás parámetros del modelo para esta celda no cambian.

Para la celda (1,k-1) también se define un vecindario y comportamiento diferente al resto. Aquí:

$\eta = 4$

N = { (1,-1), (1,0), (0,0), (0,-1) }

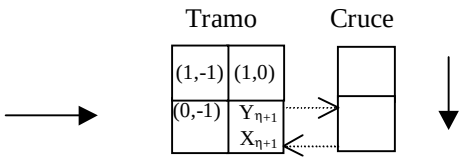


Figura 12 – Vecindario y acoplamiento de la celda (1,k-1)

La función τ y el tipo de demora (delay) para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports	Delay	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	send(0, y-c-hayauto)	transport	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (1,0) = 1 and (1,-1) = 1)	send(0, y-c-hayauto)	transport	Llega_Desde_CarrilIz_Con Prioridad
0	((0,0) = 1 and portvalue(x-c-haylugar) = 0)	send(1, y-c-hayauto)	inercial	Sale_Hacia_Cruce
(0,0)	t /*en otro caso conserva estado */	send(0, y-c-hayauto)	transport	Default

Las reglas para avanzar hacia esta celda son las mismas que las definidas para las celdas del carril 1; mientras que para abandonar la celda, se debe ingresar al cruce. Los demás parámetros del modelo para esta celda no cambian.

La utilización de los ports externos y la regla de avance de autos hacia el cruce (Sale_Hacia_Cruce) para las celdas de la columna k-1 es la misma que se ha definido para la celda (0,k-1) del tramo de carril único (sección 1.3.1.1).

1.3.1.3. Calles de tres carriles de igual dirección

Un tramo t = (p1, p2, 3, a, dir, max) de 3 carriles se define como un modelo Cell_DEVS de dos dimensiones con demora de transporte, cuya estructura se presenta en la Figura 13.

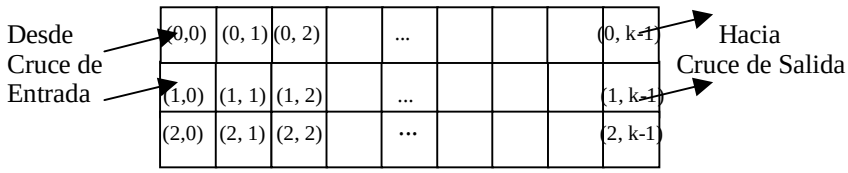


Figura 13 – Tramo de 3 carriles

Las celdas de cada fila de este espacio tienen comportamiento distinto, determinado por la conformación de su vecindario, y serán especificadas por separado. Es decir, los vehículos de las celdas de la primera fila sólo se pueden mover hacia adelante y en diagonal derecha; los de la segunda fila lo pueden hacer hacia adelante y en ambas diagonales; y los de la última fila se pueden mover hacia adelante y en diagonal izquierda.

Las celdas de la primera fila (carril 0) del espacio se definen como:

$$C_{0j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = < η, P^x, P^y>, donde

η = 8;

P^x = { (X_1, binario), (X_2, binario), (X_3, binario), (X_4, binario), (X_5, binario), (X_6, binario), (X_7, binario), (X_8, binario) };

P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario), (Y_4, binario), (Y_5, binario) , (Y_6, binario), (Y_7, binario), (Y_8, binario) }.

X = ***Y*** = {0, 1};

S:

$$s = \begin{cases} 1 \text{ si hay un vehículo en la celda;} \\ 0 \text{ sino.} \end{cases}$$

N = { (0,0), (0,1), (0,-1), (-1,1), (-1,-1), (-1,0), (-2,0), (-2,1) };

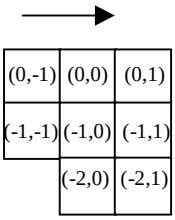


Figura 14 - Vecindario de la celda origen (carril 0)

delay = transport;

d = Conversión_Demora(velocidad(max));

λ, **δ_{int}** y **δ_{ext}** se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

τ: S x N → S se define de la siguiente manera,

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (-1,-1) = 1 and (-1,0) = 1)	Llega_Desde_CarrilDer
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (-1,1) = 0 and (-1,0) = 0 and ((-2,1) = 0 or (-2,0) = 0))	Sale_Hacia_CarrilDer_ SinPrioridad
(0,0)	t /*en otro caso conserva estado */	Default

Los autos que avanzan hacia estas celdas sólo lo pueden hacer desde la celda de atrás o desde el otro carril (derecho), debido a la conformación del vecindario. Luego, los autos que abandonan estas celdas lo pueden hacer hacia adelante o en diagonal derecha. Para la regla Sale_Hacia_CarrilDer_SinPrioridad se tiene en cuenta que los autos que avanzan derecho tienen prioridad, luego los que se mueven hacia izquierda y por último los que lo hacen hacia derecha. Por esto cuando un auto quiere avanzar hacia derecha debe verificar que no se interponga en el camino de otro vehículo. Para esta regla hay 2 movimientos con prioridad que deben ser chequeados: el avance derecho sobre el carril de destino ((-1,0) = 0) y avance desde derecha hacia el carril destino ((-2,1) = 0 or (-2,0) = 0)). Las demás reglas ya fueron explicadas en modelos anteriores.

Las celdas de la segunda fila (carril 1) del espacio se definen como:

$$C_{1j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = $< \eta, P^x, P^y >$, donde

$\eta = 9$;

P^x = { (X_1, binario), (X_2, binario), (X_3, binario), (X_4, binario), (X_5, binario), (X_6, binario), (X_7, binario), (X_8, binario), (X_9, binario) };

P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario), (Y_4, binario), (Y_5, binario) , (Y_6, binario), (Y_7, binario), (Y_8, binario), (Y_9, binario) }.

X = ***Y*** = {0, 1};

S:

$$s = \begin{cases} 1 \text{ si hay un vehículo en la celda;} \\ 0 \text{ sino.} \end{cases}$$

N = { (0,0), (0,1), (1,0), (1,1), (0,-1), (1,-1), (-1,1), (-1,-1), (-1,0) };

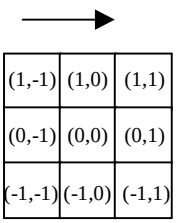


Figura 15 - Vecindario de la celda origen (carril 1)

delay = transport;

d = Conversión_Demora(velocidad(max));

λ, δ_{int} y δ_{ext} se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

τ: S x N $\rightarrow$ S se define de la siguiente manera,

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (-1,-1) = 1 and (-1,0) = 1) or	Llega_Desde_CarrilDer
	((0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1)	Llega_Desde_CarrilIz_ConPrioridad
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (1,1) = 0 and (1,0) = 0) or	Sale_Hacia_CarrilIz
	((0,0) = 1 and (-1,1) = 0 and (-1,0) = 0)	Sale_Hacia_CarrilDer_ConPrioridad
(0,0)	t /*en otro caso conserva estado */	Default

Los autos que avanzan hacia estas celdas y que salen de ellas lo pueden hacer utilizando los 3 movimientos posibles (derecho, diagonal izquierda y diagonal derecha). Todas estas reglas ya fueron explicadas en modelos anteriores.

Las celdas de la tercera fila (carril 2) del espacio se definen como:

$$C_{2j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = < η, P^x, P^y>, donde

$\eta = 8$;

P^x = { (X_1, binario), (X_2, binario), (X_3, binario), (X_4, binario), (X_5, binario), (X_6, binario), (X_7, binario), (X_8, binario) };

P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario), (Y_4, binario), (Y_5, binario) , (Y_6, binario), (Y_7, binario), (Y_8, binario) }.

X = ***Y*** = {0, 1};

S:

s = { 1 si hay un vehículo en la celda;
0 sino.

N = { (0,0), (0,1), (1,0), (1,1), (1,-1), (0,-1), (2,0), (2,-1) };

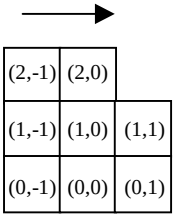


Figura 16 - Vecindario de la celda origen (carril 2)

delay = transport;

d = Conversión_Demora(velocidad(max));

λ, δ_{int} y δ_{ext} se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

τ: S x N $\rightarrow$ S se define de la siguiente manera,

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1 and ((2,-1) = 1 or (2,0) = 1))	Llega_Desde_CarrilIz_SinPrioridad
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (1,1) = 0 and (1,0) = 0)	Sale_Hacia_CarrilIz
(0,0)	t /*en otro caso conserva estado */	Default

Los autos que avanzan hacia estas celdas sólo lo pueden hacer desde la celda de atrás o desde el otro carril (izquierdo), debido a la conformación del vecindario. Para la regla Llega_Desde_CarrilIz_SinPrioridad se debe tener en cuenta que los autos primero intentan avanzar derecho, luego hacia izquierda y por último hacia derecha. Por esto, un auto avanza desde el carril izquierdo hacia la celda origen sólo cuando no puede hacer ninguno de los otros 2 movimientos (avanzar derecho y avanzar hacia la izquierda). Luego, los autos que abandonan estas celdas lo pueden hacer hacia adelante o hacia la diagonal izquierda. Las demás reglas ya fueron explicadas en modelos anteriores.

El modelo acoplado correspondiente al tramo t = (p1, p2, 3, a, dir, max) se define como:

$$TC3(k, max) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z, select >$$

donde,

Ylist = { (i,0) / $0 \le i < 3$ } U { (i, k-1) / $0 \le i < 3$ }

Xlist = { (i,0) / $0 \le i < 3$ } U { (i, k-1) / $0 \le i < 3$ }

I = $<P^x, P^y>$

$P^x = \{<X_{\eta+1}(i,0), binario> / \ 0 \le i < 3\} \cup \{<X_{\eta+1}(i,k-1), binario> / \ 0 \le i < 3\}$
$P^y = \{<Y_{\eta+1}(i,0), binario> / 0 \le i < 3\} \cup \{<Y_{\eta+1}(i,k-1), binario> / \ 0 \le i < 3\}$

Estos ports serán denotados de la siguiente forma:

Port	Nombre
$X_{\eta+1}(i,0)$, $0 \le i \le 2$	x-c-hayauto
$X_{\eta+1}(i,k-1)$, $0 \le i \le 2$	x-c-haylugar
$Y_{\eta+1}(i,0)$, $0 \le i \le 2$	y-c-haylugar
$Y_{\eta+1}(i,k-1)$, $0 \le i \le 2$	y-c-hayauto

$\boldsymbol{X}$ = { 0, 1 }

$\boldsymbol{Y}$ = { 0, 1 }

n = 2

$\mathbf{t_1}$ = 3

$\mathbf{t_2}$ = k

N y η han sido descriptos al definir el modelo de celda de cada carril.

C = { C_{ij} / i ∈ [0, 2] ∧ j ∈ [0, k-1] }, donde cada C_{ij} es un componente Cell-DEVS con demora y la especificación de cada celda ha sido descripta antes de introducir el modelo acoplado.

B = { (0,0), (1,0), (2,0), (0,k-1), (1,k-1), (2,k-1) }

Z se construye siguiendo la definición dada por el formalismo Cell_DEVS, instanciada con el vecindario de este espacio.

select = { (0,1), (1,1), (-1,1), (0,0), (1,0), (-1,0), (1,-1), (0,-1), (-1,-1) };

TC3(k, max) significa Tramo de 3 Carriles de longitud k (celdas) cada uno, con velocidad máxima max (en Km/h). Donde k = Ctd_Celdas(t) y max es la constante especificada en t.

El modelo para 3 carriles es una extensión del tramo de un sólo carril, permitiendo que los vehículos además de moverse a la celda de adelante, lo puedan hacer hacia la que tienen adelante en cualquiera de las diagonales. Cada tramo se extiende entre un cruce de origen y otro de destino, por lo tanto se definen los ports que acoplan estos modelos y permiten el avance de tráfico de uno al otro. La interfaz de este modelo la conforman las celdas de la primera y última columna del espacio, pues cada una debe intercambiar información con los modelos de los cruces correspondientes.

El comportamiento para las celdas borde es distinto al definido anteriormente. Para la celda (0,0) se define el siguiente comportamiento y vecindario:

η = 6

N = { (0,0), (0,1), (-1,1), (-1,0), (-2,0), (-2,1) };

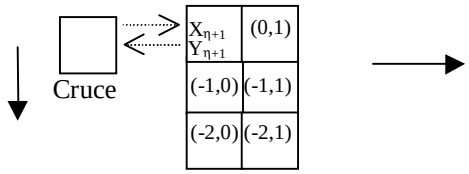


Figura 17 – Vecindario y acoplamiento de la celda (0,0)

La función τ para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nombre de la regla

1	((0,0) = 0 and portvalue(x-c-hayauto) = 1)	Llega_Desde_Cruce
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (-1,1) = 0 and (-1,0) = 0 and ((-2,1) = 0 or (-2,0) = 0))	Sale_Hacia_CarrilDer_SinPrioridad
(0,0)	t /*en otro caso conserva estado */	Default

Esta celda sólo recibe vehículos que salen del cruce, mientras que las reglas para avanzar desde ella son las mismas que las definidas para las celdas del carril 0. Los demás parámetros del modelo para esta celda no cambian.

El comportamiento y vecindad de la celda (1,0) se define como:

$\eta = 6$

N = { (0,0), (0,1), (-1,1), (-1,0), (1,0), (1,1) };

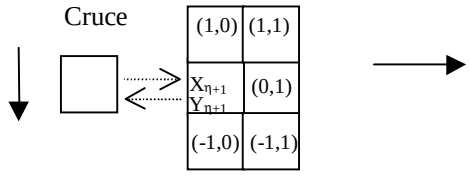


Figura 18 – Vecindario y acoplamiento de la celda (1,0)

La función τ para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and portvalue(x-c-hayauto) = 1)	Llega_Desde_Cruce
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (1,1) = 0 and (1,0) = 0) or	Sale_Hacia_CarrilIz
	((0,0) = 1 and (-1,1) = 0 and (-1,0) = 0)	Sale_Hacia_CarrilDer_ ConPrioridad
(0,0)	t /*en otro caso conserva estado */	Default

Esta celda sólo recibe vehículos que salen del cruce, mientras que las reglas para avanzar desde ella son las mismas que las definidas para las celdas del carril 1. Los demás parámetros del modelo para esta celda no cambian.

Convención 4

Cuando se define el comportamiento de una celda o conjunto de celdas C, en forma análoga a un subespacio S de otro modelo, significa que los modelos son iguales a excepción del parámetro que representa la velocidad máxima. Es decir, la demora para las celdas de C se calcula utilizando la velocidad máxima establecida para el modelo al que pertenecen; mientras que las celdas de S pueden tener otra diferente.

El comportamiento y vecindad de la celda (2,0) se define en forma análoga al de la celda (1,0) del tramo de 2 carriles (sección 1.3.1.2).

La utilización de los ports externos y la regla de avance de autos desde el cruce (Llega_Desde_Cruce) para las celdas de la columna 0 es la misma que se ha definido para la celda (0,0) del tramo de carril único (sección 1.3.1.1).

El comportamiento y vecindad de la celda (0,k-1) se define en forma análoga al de la celda (0,k-1) del tramo de 2 carriles (sección 1.3.1.2).

El comportamiento y vecindad de la celda (1,k-1) se define como:

$\eta = 6$

N = { (0,0), (0,-1), (-1,-1), (-1,0), (1,0), (1,-1) };

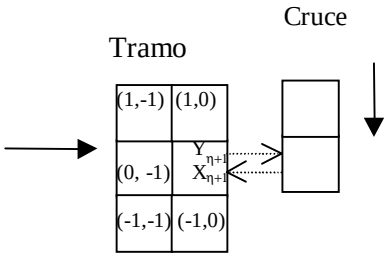


Figura 19 – Vecindario y acoplamiento de la celda (1,k-1)

La función τ y el tipo de demora (delay) para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports	Delay	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	send(0, y-c-hayauto)	transport	Llega_Desde_Atrás
	((0,0) = 0 and (-1,-1) = 1 and (-1,0) = 1 and (0,-1) = 0) or	send(0, y-c-hayauto)	transport	Llega_Desde_Carril Der
	((0,0) = 0 and (1,0) = 1 and (1,-1) = 1 and (0,-1) = 0)	send(0, y-c-hayauto)	transport	Llega_Desde_Carril Iz_ConPrioridad
0	((0,0) = 1 and portvalue(x-c-haylugar) = 0)	send(1, y-c-hayauto)	inercial	Sale_Hacia_Cruce
(0,0)	t /*en otro caso conserva estado */	send(0, y-c-hayauto)	transport	Default

Las reglas para avanzar hacia esta celda son las mismas que las definidas para las celdas del carril 1; mientras que para abandonar la celda, se debe ingresar al cruce. Los demás parámetros del modelo para esta celda no cambian.

El comportamiento y vecindad de la celda (2,k-1) se define como:

$\eta = 6$

N = { (0,0), (0,-1), (1,-1), (1,0), (2,0), (2,-1) };

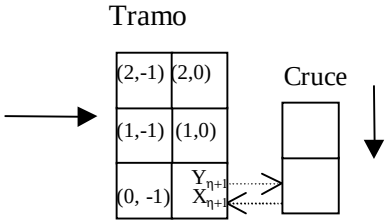


Figura 20 – Vecindario y acoplamiento de la celda (2,k-1)

La función τ para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports	Delay	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	send(0, y-c-hayauto)	transport	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1 and ((2,0) = 1 or (2,-1) = 1))	send(0, y-c-hayauto)	transport	Llega_Desde_CarrilIz_SinPrioridad
0	((0,0) = 1 and portvalue(x-c-haylugar) = 0)	send(1, y-c-hayauto)	inercial	Sale_Hacia_Cruce
(0,0)	t /*en otro caso conserva estado */	send(0, y-c-hayauto)	transport	Default

Las reglas para avanzar hacia esta celda son las mismas que las definidas para las celdas del carril 2; mientras que para abandonar la celda, se debe ingresar al cruce. Los demás parámetros del modelo para esta celda no cambian.

La utilización de los ports externos y la regla de avance de autos hacia el cruce (Sale_Hacia_Cruce) para las celdas de la columna k-1 es la misma que se ha definido para la celda (0,k-1) del tramo de carril único (sección 1.3.1.1).

1.3.1.4. Calles de cuatro carriles de igual dirección

Un tramo t = (p1, p2, 4, a, dir, max) de 4 carriles se define como un modelo Cell_DEVS de dos dimensiones con demora de transporte, cuya estructura se presenta en la Figura 21.

Figura 21 – Tramo de 4 carriles

Las celdas de cada fila de este espacio tienen comportamiento distinto, determinado por la conformación de su vecindario, y serán especificadas por separado. Las celdas de la primera fila

se comportan en forma análoga a las celdas del carril 0 del modelo de 3 carriles, lo mismo ocurre para las celdas del carril 3 que se definen como las del carril 2 de TC3 (sección 1.3.1.3). A continuación se especifican las celdas de los carriles 1 y 2 para el tramo de 4 carriles.

Las celdas de la segunda fila (carril 1) del espacio se definen como:

$$C_{1j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = < η, P^x, P^y>, donde

η = 11;

P^x = { (X_1, binario), (X_2, binario), (X_3, binario), (X_4, binario), (X_5, binario), (X_6, binario), (X_7, binario), (X_8, binario), (X_9, binario), (X_{10}, binario), (X_{11}, binario) };

P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario), (Y_4, binario), (Y_5, binario) , (Y_6, binario), (Y_7, binario), (Y_8, binario), (Y_9, binario), (Y_{10}, binario), (Y_{11}, binario) }.

X = ***Y*** = {0, 1};

S:

s = { 1 si hay un vehículo en la celda; 0 sino.

N = { (1,1), (1,-1), (1,0), (0,0), (0,1), (0,-1), (-1,1), (-1,-1), (-1,0), (-2,0), (-2,1) };

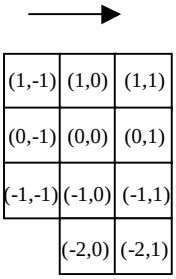


Figura 22 - Vecindario de la celda origen (carril 1)

delay = transport;

d = Conversión_Demora(velocidad(max));

λ, **δ_{int}** y **δ_{ext}** se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

τ: S x N $\rightarrow$ S se define de la siguiente manera,

Nuevo Estado	Estado del vecindario	Nombre de la regla

1	((0,0) = 0 and (0,-1) = 1) or	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (-1,-1) = 1 and (-1,0) = 1) or	Llega_Desde_CarrilDer
	((0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1)	Llega_Desde_CarrilIz_Con Prioridad
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (1,1) = 0 and (1,0) = 0) or	Sale_Hacia_CarrilIz
	((0,0) = 1 and (-1,1) = 0 and (-1,0) = 0 and ((-2,1) = 0 or (-2,0) = 0))	Sale_Hacia_CarrilDer_Sin Prioridad
(0,0)	t /*en otro caso conserva estado */	Default

Los autos que avanzan hacia estas celdas y que salen de ellas lo pueden hacer utilizando los 3 movimientos posibles (derecho, diagonal izquierda y diagonal derecha). Todas estas reglas ya fueron explicadas en modelos anteriores.

Las celdas de la tercera fila (carril 2) del espacio se definen como:

$$C_{2j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = $< \eta, P^x, P^y >$, donde

$\eta = 11$;

P^x = { (X_1, binario), (X_2, binario), (X_3, binario), (X_4, binario), (X_5, binario), (X_6, binario), (X_7, binario), (X_8, binario), (X_9, binario), (X_{10}, binario), (X_{11}, binario) };

P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario), (Y_4, binario), (Y_5, binario) , (Y_6, binario), (Y_7, binario), (Y_8, binario), (Y_9, binario), (Y_{10}, binario), (Y_{11}, binario) }.

X = ***Y*** = {0, 1};

S:

$$s = \begin{cases} 1 \text{ si hay un vehículo en la celda;} \\ 0 \text{ sino.} \end{cases}$$

N = { (1,1), (1,-1), (1,0), (0,0), (0,1), (0,-1), (-1,1), (-1,-1), (-1,0), (2,-1), (2,0) };

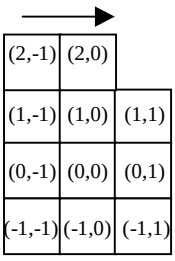


Figura 23 - Vecindario de la celda origen (carril 2)

delay = transport;

d = Conversión_Demora(velocidad(max));

λ, δ_{int} y δ_{ext} se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

τ: S x N $\rightarrow$ S se define de la siguiente manera,

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (-1,-1) = 1 and (-1,0) = 1) or	Llega_Desde_CarrilDer
	((0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1 and ((2,-1) = 1 or (2,0) = 1))	Llega_Desde_CarrilIz_Sin Prioridad
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (1,1) = 0 and (1,0) = 0) or	Sale_Hacia_CarrilIz
	((0,0) = 1 and (-1,1) = 0 and (-1,0) = 0)	Sale_Hacia_CarrilDer_Con Prioridad
(0,0)	t /*en otro caso conserva estado */	Default

Los autos que avanzan hacia estas celdas y que salen de ellas lo pueden hacer utilizando los 3 movimientos posibles (derecho, diagonal izquierda y diagonal derecha). Todas estas reglas ya fueron explicadas en modelos anteriores.

El modelo acoplado correspondiente al tramo t = (p1, p2, 4, a, dir, max) se define como:

$$TC4(k, max) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z, select >$$

donde,
Ylist = { (i,0) / $0 \le i < 4$ } U { (i, k-1) / $0 \le i < 4$ }

Xlist = { (i,0) / $0 \le i < 4$ } U { (i, k-1) / $0 \le i < 4$ }

I = $<P^x, P^y>$

P^x = {$<X_{\eta+1}(i,0)$, binario> / $0 \le i < 4$} U {$<X_{\eta+1}(i,k-1)$, binario> / $0 \le i < 4$}
P^y = {$<Y_{\eta+1}(i,0)$, binario> / $0 \le i < 4$} U {$<Y_{\eta+1}(i,k-1)$, binario> / $0 \le i < 4$}

Estos ports serán denotados de la siguiente forma:

Port	Nombre
$X_{\eta+1}(i,0)$, $0 \le i \le 3$	x-c-hayauto
$X_{\eta+1}(i,k-1)$, $0 \le i \le 3$	x-c-haylugar
$Y_{\eta+1}(i,0)$, $0 \le i \le 3$	y-c-haylugar
$Y_{\eta+1}(i,k-1)$, $0 \le i \le 3$	y-c-hayauto

X = { 0, 1 }

Y = { 0, 1 }

n = 2

$\mathbf{t_1}$ = 4

$\mathbf{t_2}$ = k

N y η han sido descriptos al definir el modelo de celda de cada carril.

C = { C_{ij} / i ∈ [0, 3] ∧ j ∈ [0, k-1] }, donde cada C_{ij} es un componente Cell-DEVS con demora y la especificación de cada celda ha sido descripta antes de introducir el modelo acoplado.

B = { (0,0), (1,0), (2,0), (3,0), (0,k-1), (1,k-1), (2,k-1), (3,k-1) }

Z se construye siguiendo la definición dada por el formalismo Cell_DEVS, instanciada con el vecindario de este espacio.

select = { (0,1), (1,1), (-1,1), (0,0), (1,0), (-1,0), (1,-1), (0,-1), (-1,-1) };

TC4(k, max) significa Tramo de 4 Carriles de longitud k (celdas) cada uno, con velocidad máxima max (en Km/h). Donde k = Ctd_Celdas(t) y max es la constante especificada en t.

El modelo para 4 carriles es una extensión del tramo de 3, que incorpora una línea más de tráfico provocando que las condiciones para avanzar de algunas celdas difieran. Cada tramo se extiende entre un cruce de origen y otro de destino, por lo tanto se definen los ports que acoplan estos modelos y permiten el avance de tráfico de uno al otro. La interfaz de este modelo la conforman las celdas de la primera y última columna del espacio, pues cada una debe intercambiar información con los modelos de los cruces correspondientes.

El comportamiento para las celdas borde es distinto al definido anteriormente. El comportamiento y vecindad de la celda (0,0) se define en forma análoga a los establecidos para la celda (0,0) del modelo de 3 carriles (sección 1.3.1.3).

El comportamiento y vecindad de la celda (1,0) se define como:

η = 8

N = { (0,0), (0,1), (1,0), (1,1), (-1,0), (-1,1), (-2, 0), (-2,1) }

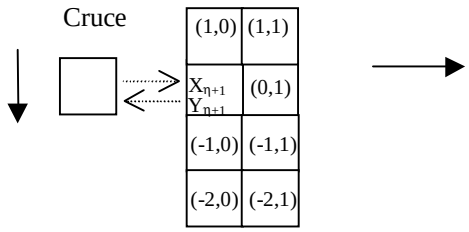


Figura 24 – Vecindario y acoplamiento de la celda (1,0)

La función τ para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nombre de la regla

1	((0,0) = 0 and portvalue(x-c-hayauto) = 1)	Llega_Desde_Cruce
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (1,0) = 0 and (1,1) = 0) or	Sale_Hacia_CarrilIz
	((0,0) = 1 and (-1,0) = 0 and (-1,1) = 0 and ((-2,1) = 0 or (-2,0) = 0))	Sale_Hacia_CarrilDer_SinPrioridad
(0,0)	t /*en otro caso conserva estado */	Default

Esta celda sólo recibe vehículos que salen del cruce, mientras que las reglas para avanzar desde ella son las mismas que las definidas para las celdas del carril 1. Los demás parámetros del modelo para esta celda no cambian.

El comportamiento y vecindad de la celda (2,0) se define en forma análoga a los establecidos para la celda (1,0) del modelo de 3 carriles (sección 1.3.1.3).

El comportamiento y vecindad de la celda (3,0) se define en forma análoga a los establecidos para la celda (2,0) del modelo de 3 carriles (sección 1.3.1.3).

La utilización de los ports externos y la regla de avance de autos desde el cruce (Llega_Desde_Cruce) para las celdas de la columna 0 es la misma que se ha definido para la celda (0,0) del tramo de carril único (sección 1.3.1.1).

El comportamiento y vecindad de la celda (0,k-1) se define en forma análoga a los establecidos para la celda (0,k-1) del modelo de 3 carriles (sección 1.3.1.3).

El comportamiento y vecindad de la celda (1,k-1) se define en forma análoga a los establecidos para la celda (1,k-1) del modelo de 3 carriles (sección 1.3.1.3).

El comportamiento y vecindad de la celda (2,k-1) se define como:

$\eta = 8$

N = { (0,0), (0,-1), (1,0), (1,-1), (-1,0), (-1,-1), (2,0), (2,-1) }

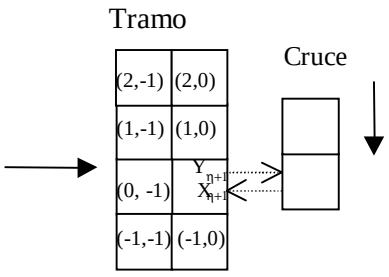


Figura 25 – Vecindario y acoplamiento de la celda (2,k-1)

La función τ y el tipo de demora (delay) para esta celda se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports	Delay	Nombre de la regla

1	((0,-1) = 1 and (0,0) = 0) or	send(0, y-c-hayauto)	transport	Llega_Desde_Atrás
	((0,0) = 0 and (1,0) = 1 and (1,-1) = 1 and (0,-1) = 0 and ((2,-1) = 1 or (2,0) = 1)) or	send(0, y-c-hayauto)	transport	Llega_Desde_CarrilIz_SinPrioridad
	((0,0) = 0 and (-1,-1) = 1 and (-1,0) = 1 and (0,-1) = 0)	send(0, y-c-hayauto)	transport	Llega_Desde_CarrilDer
0	((0,0) = 1 and portvalue(x-c-haylugar) = 0)	send(1, y-c-hayauto)	inercial	Sale_Hacia_Cruce
(0,0)	t /* en otro caso conserva estado */	send(0, y-c-hayauto)	transport	Default

Las reglas para avanzar hacia esta celda son las mismas que las definidas para las celdas del carril 2; mientras que para abandonar la celda, se debe ingresar al cruce. Los demás parámetros del modelo para esta celda no cambian.

El comportamiento y vecindad de la celda (3,k-1) se define en forma análoga a los establecidos para la celda (2,k-1) del modelo de 3 carriles (sección 1.3.1.3).

La utilización de los ports externos y la regla de avance de autos hacia el cruce (Sale_Hacia_Cruce) para las celdas de la columna k-1 es la misma que se ha definido para la celda (0,k-1) del tramo de carril único (sección 1.3.1.1).

1.3.1.5. Calles con más de cuatro carriles de igual dirección

Un tramo t = (p1, p2, #c, a, dir, max) con más de 4 carriles (#c > 4) se define como un modelo Cell_DEVS de dos dimensiones con demora de transporte, cuya estructura se presenta en la Figura 26.

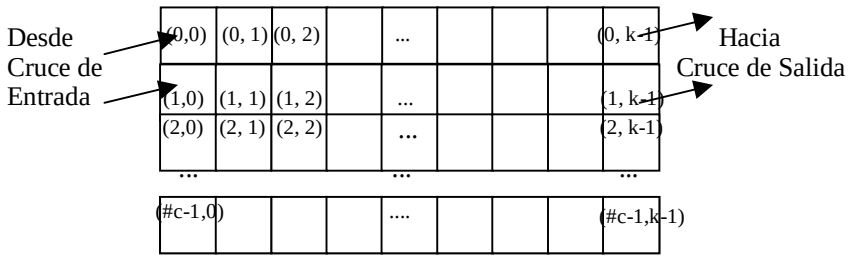


Figura 26 – Tramo con más de 4 carriles

Las celdas de cada fila de este espacio tienen comportamiento distinto, determinado por la conformación de su vecindario, y serán especificadas por separado. Las celdas del carril 0 se definen en forma análoga a las celdas del carril 0 del modelo de 4 carriles. Lo mismo ocurre para las celdas del carril 1 que se definen como las del carril 1 de TC4. Las celdas del carril #c-2 se definen en forma análoga a las celdas del carril 3 del modelo de 4 carriles. Lo mismo ocurre para las celdas del carril #c-1 que se definen como las del carril 4 de TC4 (TC4 está especificado en la sección 1.3.1.4).

A continuación se especifican las celdas del conjunto {(i,j) / $1 < i < \#c-2 \wedge 0 < j < k-1$}, para tramos de más 4 carriles:

$$C_{ij} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = < η, P^x, P^y>, donde

η = 13;

P^x = { (X_1, binario), (X_2, binario), (X_3, binario), (X_4, binario), (X_5, binario), (X_6, binario), (X_7, binario), (X_8, binario), (X_9, binario), (X_{10}, binario), (X_{11}, binario), (X_{12}, binario), (X_{13}, binario) };

P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario), (Y_4, binario), (Y_5, binario) , (Y_6, binario), (Y_7, binario), (Y_8, binario), (Y_9, binario), (Y_{10}, binario), (Y_{11}, binario), (Y_{12}, binario), (Y_{13}, binario) }.

X = ***Y*** = {0, 1};

S:

s = { 1 si hay un vehículo en la celda;
0 sino.

N = { (0,0), (0,1), (1,0), (1,1), (0,-1), (1,-1), (-1,1), (-1,-1), (-1,0), (2,-1), (2,0), (-2,0), (-2,1) };

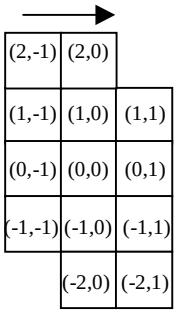


Figura 27 - Vecindario de la celda origen

delay = transport;

d = Conversión_Demora(velocidad(max));

λ, **δ_{int}** y **δ_{ext}** se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

τ: S x N → S se define de la siguiente manera,

Nuevo Estado	Estado del vecindario	Nombre de la regla
1	((0,0) = 0 and (0,-1) = 1) or	Llega_Desde_Atrás
	((0,0) = 0 and (0,-1) = 0 and (-1,-1) = 1 and (-1,0) = 1) or	Llega_Desde_CarrilDer

	((0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1 and ((2,-1) = 1 or (2,0) = 1))	Llega_Desde_CarrilIz_SinPrioridad
0	((0,0) = 1 and (0,1) = 0) or	Sale_Hacia_Adelante
	((0,0) = 1 and (1,1) = 0 and (1,0) = 0) or	Sale_Hacia_CarrilIz
	((0,0) = 1 and (-1,1) = 0 and (-1,0) = 0 and ((-2,1) = 0 or (-2,0) = 0))	Sale_Hacia_CarrilDer_SinPrioridad
(0,0)	t /*en otro caso conserva estado */	Default

Los autos que avanzan hacia estas celdas y que salen de ellas lo pueden hacer utilizando los 3 movimientos posibles (derecho, diagonal izquierda y diagonal derecha). Todas estas reglas ya fueron explicadas en modelos anteriores.

El modelo acoplado correspondiente al tramo t = (p1, p2, #c, a, dir, max) se define como:

$$TC(k, max, \#c) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z, select >$$

donde,

Ylist = { (i,0) / 0 ≤ i < #c } U { (i, k-1) / 0 ≤ i < #c }

Xlist = { (i,0) / 0 ≤ i < #c } U { (i, k-1) / 0 ≤ i < #c }

I = $<P^x, P^y>$

$P^x = \{<X_{\eta+1}(i,0), binario> / \ 0 \le i < \#c\} \cup \{<X_{\eta+1}(i,k-1), binario> / \ 0 \le i < \#c\}$
$P^y = \{<Y_{\eta+1}(i,0), binario> / 0 \le i < \#c\} \cup \{<Y_{\eta+1}(i,k-1), binario> / \ 0 \le i < \#c\}$

Estos ports serán denotados de la siguiente forma:

Port	Nombre
$X_{\eta+1}(i,0)$, 0 ≤ i ≤ #c-1	x-c-hayauto
$X_{\eta+1}(i,k-1)$, 0 ≤ i ≤ #c-1	x-c-haylugar
$Y_{\eta+1}(i,0)$, 0 ≤ i ≤ #c-1	y-c-haylugar
$Y_{\eta+1}(i,k-1)$, 0 ≤ i ≤ #c-1	y-c-hayauto

X = { 0, 1 }

Y = { 0, 1 }

n = 2

$\mathbf{t_1}$ = #c

$\mathbf{t_2}$ = k

N y **η** han sido descriptos al definir el modelo de celda de cada carril.

C = { C_{ij} / i ∈ [0, #c-1] ∧ j ∈ [0, k-1] }, donde cada C_{ij} es un componente Cell-DEVS con demora y la especificación de cada celda ha sido descripta antes de introducir el modelo acoplado.

B = { (i, 0) / 0 ≤ i ≤ #c-1 } U { (i, k-1) / 0 ≤ i ≤ #c-1 }

Z se construye siguiendo la definición dada por el formalismo Cell_DEVS, instanciada con el vecindario de este espacio.

select = { (0,1), (1,1), (-1,1), (0,0), (1,0), (-1,0), (1,-1), (0,-1), (-1,-1) };

TC (k, max, #c) significa Tramo de #c Carriles de longitud k (celdas) cada uno, con velocidad máxima max (en Km/h). Donde max y #c es la velocidad máxima y cantidad de carriles especificada para t, respectivamente; y k = Ctd_Celdas(t).

El comportamiento para las celdas borde es distinto al definido anteriormente. El comportamiento y vecindad de la celda (0,0) se define en forma análoga a los establecidos para la celda (0,0) del modelo de 4 carriles (sección 1.3.1.4).

El comportamiento y vecindad de la celda (#c-2,0) se define en forma análoga a los establecidos para la celda (2,0) del modelo de 4 carriles (sección 1.3.1.4).

El comportamiento y vecindad de la celda (#c-1,0) se define en forma análoga a los establecidos para la celda (3,0) del modelo de 4 carriles (sección 1.3.1.4).

El comportamiento y vecindad de la celda (0,k-1) se define en forma análoga a los establecidos para la celda (0,k-1) del modelo de 4 carriles (sección 1.3.1.4).

El comportamiento y vecindad de la celda (1,k-1) se define en forma análoga a los establecidos para la celda (1,k-1) del modelo de 4 carriles (sección 1.3.1.4).

El comportamiento y vecindad de la celda (#c-1,k-1) se define en forma análoga a los establecidos para la celda (3,k-1) del modelo de 4 carriles (sección 1.3.1.4).

El comportamiento y vecindad de las celdas { (j, 0) / 0 < j < #c-2 } se define en forma análoga a los establecidos para la celda (1,0) del modelo de 4 carriles (sección 1.3.1.4).

El comportamiento y vecindad de las celdas { (j, k-1) / 1 < j < #c-1 } se define en forma análoga a los establecidos para la celda (2,k-1) del modelo de 4 carriles (sección 1.3.1.4).

1.3.1.6. Calles con carriles de distinta dirección

Para definir una calle con carriles de distinta dirección se deben definir 2 tramos entre el mismo par de cruces pero con dirección opuesta. Por ejemplo, se puede definir t1 = (p1, p2, n, a, 1, max) y t2 = (p1, p2, n, a, 0, max). De esta forma cada tramo mapea en un Cell-DEVS como los definidos en las secciones 1.3.1.1 - 1.3.1.5.

La definición de las calles doble mano como dos tramos independientes produce como restricción que un vehículo que circula en una dirección no pueda utilizar el carril contrario. Este tipo de maniobras es válido en las rutas y por lo tanto no representa una limitación para el modelo que busca reflejar el tráfico urbano. Otro caso donde se utiliza el carril contrario es en algunos giros (por ejemplo, en avenidas doble mano para girar a la izquierda), pero esto sí se

encuentra modelado dentro de la estructura de los cruces, es decir la restricción sólo afecta a los tramos.

1.3.2. Cruces

El conjunto de los cruces se obtiene a partir de los tramos definidos como:

$$\text{Cruces} = \{ (c, maxc) / maxc \in N \wedge \exists\, t, t' \in \text{Tramos} \wedge t = (p1, p2, n, a, dir, max) \wedge t' = (p1', p2', n', a', dir', max') \wedge t \neq t' \wedge (p1 = c \vee p2 = c) \wedge (p1' = c \vee p2' = c) \}$$

Los elementos de este conjunto se definen como puntos del espacio de dos dimensiones que representan los lugares donde se cruzan 2 ó más tramos, asociados con su velocidad máxima de circulación permitida. Pero un cruce siempre debe tener por lo menos un tramo de ingreso y uno de salida, pues si esto no sucede en realidad no se trata de una intersección de calles sino del lugar donde nacen o terminan, lo cual será modelado en la sección 1.4.8 al describir el marco experimental. Esta restricción se escribe como:

$$\forall\, (c, maxc) \in \text{Cruces}:$$
$$\exists\, t, t' \in \text{Tramos} \wedge t = (p1, p2, n, a, dir, max) \wedge t' = (p1', p2', n', a', dir', max') \wedge t \neq t' \wedge$$
$$[\, (p1 = p1' = c \wedge dir \neq dir') \vee (p1 = p2' = c \wedge dir = dir') \vee (p2 = p2' = c \wedge dir \neq dir') \,]$$

De esta forma los vehículos que ingresan al cruce por algún tramo siempre tendrán por lo menos una salida disponible.

Las intersecciones se representan como un anillo de celdas que se acopla con carriles de tramos, siguiendo la propuesta que fuera planteada en [CQL95], [CLQ96] y [CDL97], y resumida en la sección 3.2 del capítulo I. Las reglas de comportamiento de los vehículos establecen que un auto dentro de la intersección (en el anillo) tiene prioridad para acceder a una posición sobre cualquier otro vehículo que esté fuera de ella. Dentro del cruce, un vehículo gira en sentido contrario a las agujas del reloj o sale, es decir no se permite que un auto se detenga a la espera de una salida particular porque esto puede provocar que en algún momento, nadie se pueda mover por estar esperando que otro vacíe la posición (deadlock). Cada vehículo avanza hacia una celda vacía, que puede estar dentro de la intersección o ser la primera de un carril de salida del cruce. Para modelar la elección del enlace de salida, se utiliza una función aleatoria local a la celda. Cada vez que un auto pasa por una salida, si hay lugar en el tramo, el vehículo chequea el valor que devuelve esta función para saber si debe seguir girando o salir. Caso contrario, el vehículo permanecerá en el cruce.

Cada cruce (c, maxc) $\in$ Cruces, se define como un modelo Cell_DEVS de una dimensión con demora de transporte y bordes conectados, cuya estructura se presenta en la Figura 28.

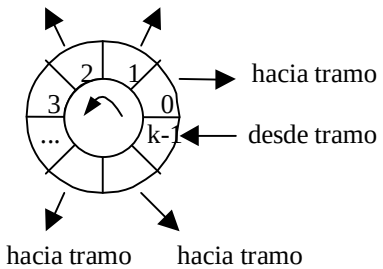


Figura 28 - Cruce

Cada celda del espacio se define como:

$$C_{0j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = < η, P^x, P^y>, donde

$\eta = 3$;
P^x = { (X_1, binario), (X_2, binario), (X_3, binario) };
P^y = { (Y_1, binario), (Y_2, binario), (Y_3, binario) }.

X = ***Y*** = {0, 1};

S:

$$s = \begin{cases} 1 \text{ si hay un vehículo en la celda;} \\ 0 \text{ sino.} \end{cases}$$

N = { (0,-1), (0,0), (0,1) };

delay = transport;

d = Conversión_Demora(velocidad(maxc));
donde, maxc es la constante especificada para el cruce y representa la velocidad máxima de circulación permitida (en km/h).

λ, δ_{int} y δ_{ext} se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

La definición de la función τ se realizará luego de introducir el modelo acoplado porque todas las celdas utilizan la información de los ports para calcular su nuevo estado.

El estado de una celda representa la presencia (s = 1) o ausencia (s = 0) de un vehículo. Dentro del cruce sólo se permite que los vehículos avancen hacia la posición de adelante, siempre y cuando la misma esté vacía. De esta forma el vecindario necesario para establecer el nuevo estado de una celda está constituido por ella misma, la celda anterior y la siguiente.

Las demoras de transporte se utilizan para modelar las demoras provocadas por la aceleración de los automotores. El movimiento de los mismos se demora antes del siguiente movimiento a la próxima celda. Este valor es generado mediante una función aleatoria que permite modelar distintas velocidades para cada vehículo en distintos momentos.

El modelo acoplado correspondiente al cruce (c, maxc) se define como:

$$Cruce(k, In, Out, maxc) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z, select >$$

donde,

Ylist = { (0,i) / $0 \leq i < k$}

Xlist = { (0,i) / $0 \leq i < k$}

I = <P^x, P^y>

P^x = {<$X_{\eta+1}(0,i)$, binario> / $0 \leq i < k$ }
P^y = {<$Y_{\eta+1}(0,i)$, binario> / $0 \leq i < k$ }

Estos ports serán denotados de la siguiente forma:

Port	Nombre	Comentario
$X_{\eta+1}(0,i)$, $i \in$ In	x-t-hayauto	Este port se usa para saber si en el tramo existe un auto que desea ingresar al cruce (portvalue(x-t-hayauto) = 1).
$X_{\eta+1}(0,i)$, $i \in$ Out	x-t-haylugar	Este port se usa para saber si en el tramo existe lugar para que salga un auto del cruce (portvalue(x-t-haylugar) = 0).
$Y_{\eta+1}(0,i)$, $i \in$ In	y-t-haylugar	Este port informa al tramo si hay suficiente lugar en el cruce para el avance un auto.
$Y_{\eta+1}(0,i)$, $i \in$ Out	y-t-hayauto	Este port informa al tramo si hay un auto que sale desde el cruce hacia él.

X = { 0, 1 }

Y = { 0, 1 }

n = 1

$\mathbf{t_1}$ = k

N = { (0,-1), (0,0), (0,1) }

C = { C_{ij} / $i = 0 \ \wedge j \in [0, k-1]$ }, donde cada C_{ij} es un componente Cell-DEVS con demora y la especificación de cada celda ha sido descripta antes de introducir el modelo acoplado.

B = {∅}

Z se construye siguiendo la definición dada por el formalismo Cell_DEVS, instanciada con el vecindario de este espacio.

select = { (0,1), (0,0), (0,-1) }

Cruce(k, In, Out, maxc) significa que es un Cruce de longitud k (celdas) con velocidad máxima de circulación permitida de maxc (Km/h), donde las posiciones de In actúan como entradas hacia el cruce y las de Out son salidas del mismo. Los conjuntos In y Out se obtienen utilizando la función Ports_In_Out,

{I, O} = Ports_In_Out((c, maxc), Tramos)

Para establecer cuáles son las celdas de entrada y cuáles las de salida, se parte de la especificación de los tramos que lo tienen como extremo y se chequea el sentido de circulación de los vehículos en ellos. En el Apéndice A se muestra la función Ports_In_Out que realiza

dicha tarea y devuelve dos conjuntos, el primero contiene cada tramo de ingreso al cruce que está conectado al mismo con el índice de la primera celda del cruce con la que se acopla; y el otro contiene la misma información pero para los tramos de salida. Esta función establece un ordenamiento de los tramos que se acoplan al mismo cruce, de forma tal que los tramos más cercanos entre sí ocupen posiciones vecinas, esto se logra ordenándolos según el ángulo de inclinación respecto a la recta y = y1, donde c = (x1,y1). Este orden también es necesario para realizar el acoplamiento de los modelos.
De los conjuntos I y O sólo se necesitan los índices de las celdas que son entradas y las que son salidas. Esto es:

$$In = \{ i+j / \exists t = (c1, c2, n, a, dir, max) \wedge (t,i) \in I \wedge j \in [0, n-1] \}$$

$$Out = \{ i+j / \exists t = (c1, c2, n, a, dir, max) \wedge (t,i) \in O \wedge j \in [0, n-1] \}$$

Para determinar la cantidad de celdas que tiene el cruce (c, maxc), basta sumar el número de carriles que tiene cada tramo asociado al cruce pues cada celda se acopla a un carril. Es decir,

$$k = \sum_{\substack{(t,i)\in(I\cup O)\wedge \\ t=(c1,c2,n,a,dir,max)}} n \qquad = \#(In) + \#(Out)$$

La especificación de este modelo representa el movimiento de los vehículos dentro de las intersecciones de calles. Cada celda del cruce se interconecta con algún tramo, por lo tanto la interfaz del modelo queda conformada por todas ellas. Los ports definidos tienen como finalidad informar sobre el estado de la celda del cruce a la del tramo y viceversa, esto se utiliza para poder calcular su nuevo estado.

Para definir la función τ se debe tener en cuenta que el comportamiento de las celdas difiere si se trata de celdas de ingreso o de salida del cruce, por lo tanto se define por separado para cada caso.

Celdas de salida del cruce

Las celdas de salida del cruce son:

$$\{ (0,i) / i \in Out \}$$

La función τ para estas celdas se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports
1	(0,0) = 0 and (0,-1) = 1 and (portvalue(x-t-haylugar) = 1 or (portvalue(x-t-haylugar) = 0 and random < p_{salir})) /* Llega auto que permanecerá dentro del cruce */	send(0, y-t-hayauto)
0	(0,0) = 0 and (0,-1) = 1 and portvalue(x-t-haylugar) = 0 and random ≥ p_{salir} /* Llega auto que abandonará el cruce */	send(1, y-t-hayauto)
0	(0,0) = 1 and (0,1) = 0	send(0, y-t-hayauto)
(0,0)	t /*En otro caso la celda conserva el estado */	send(0, y-t-hayauto)

Aquí, p_{salir} es una constante que representa la probabilidad de que un vehículo que atraviesa la celda origen abandone el cruce en su próximo movimiento.

El valor que recibe una celda de salida del cruce por los ports externos es el estado de la celda del tramo acoplada (0 si está vacía, 1 si hay un auto). Cuando un vehículo se encuentra en un celda de salida del cruce, tiene 2 opciones para su próximo movimiento: puede salir hacia el tramo acoplado o no salir (avanza dentro del cruce). Esta elección se efectúa cuando el auto **ingresa** en la celda origen y se refleja en su nuevo estado, que es 1 si permanece en el cruce y 0

si sale hacia el tramo. También la actualización del port externo (y-t-hayauto) es diferente, toma valor 1 si decide salir y 0 en otro caso.

El comportamiento de las celdas de salida se define con 4 reglas. La primera regla representa el avance de un vehículo hacia la celda origen, modelando el caso en que el auto permanecerá dentro del cruce. Esta elección se debe a que la celda del tramo acoplado está ocupada (portvalue(x-t-haylugar) = 1) o la función aleatoria devolvió un valor menor a la probabilidad de salir. Como el auto no saldrá hacia el tramo, la celda actualiza con 0 el port y-t-hayauto. La segunda regla representa el avance de un vehículo hacia la celda origen, modelando el caso en que el auto saldrá del cruce. Para que esto sea posible la celda del tramo acoplado debe estar vacía y además la función aleatoria debe ser mayor o igual a la probabilidad de salir. La celda actualiza con 1 el port y-t-hayauto para que el tramo detecte la salida del vehículo. La tercera regla representa que el vehículo de la celda de origen avanza hacia la celda siguiente dentro del cruce; siempre y cuando esta última se encuentre vacía. Cabe destacar que en este caso la celda origen tenía estado 1, que representa que el vehículo debe avanzar hacia la celda (0,1) en su próximo movimiento (es decir, en este caso no puede salir del cruce). Por último, la cuarta regla representa que la celda conserva su estado en cualquier otro caso no contemplado en las condiciones anteriores.

Celdas de ingreso al cruce

Las celdas de ingreso al cruce son:

$$\{ (0,i) / i \in In \}$$

La función τ para estas celdas se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports
1	(0,0) = 0 and ((0,-1) = 1 or portvalue(x-t-hayauto) = 1)	send(1, y-t-haylugar)
0	(0,0) = 1 and (0,1) = 0 and (0,-1) = 0 /* No hay un auto con prioridad dentro del cruce */	send(0, y-t-haylugar)
0	(0,0) = 1 and (0,1) = 0 and (0,-1) = 1 /* Hay un auto con prioridad dentro del cruce */	send(1, y-t-haylugar)
(0,0)	t /* En otro caso la celda conserva el estado */	-

El valor que puede recibir una celda de ingreso al cruce por los ports externos es 1, si existe un vehículo que avanza desde el tramo, 0 en otro caso. El valor que envían estas celdas hacia los tramos es 1 si no hay suficiente espacio (2 celdas vacías) para que pueda ingresar un vehículo al cruce, y 0 en caso contrario. Esto significa que por el port y-t-haylugar se envía 0 sólo cuando la celda origen y la anterior están vacías, permitiendo que un auto ingrese desde el tramo sólo cuando no hay otro dentro del cruce que quiera ocupar la misma posición.

Las celdas de ingreso al cruce pueden recibir vehículos desde su vecina de atrás o desde el tramo acoplado (port x-t-hayauto). La primera regla modela que si alguna de ellas tiene un vehículo y la celda de origen está vacía entonces avanzará hacia ella, actualizando el port del tramo con 1 pues la celda está ocupada. La segunda y tercera regla representan que la celda origen se vacía si la de adelante lo está; diferenciándose en el estado de la celda de atrás. Si esta última está vacía entonces se actualiza el port con valor 0 pues hay suficiente lugar para que ingrese un vehículo desde el tramo, caso contrario ((0,-1) = 1) el estado del port se actualiza con 1 porque hay un auto dentro del cruce que tiene prioridad de acceder a la celda origen y los

coches del tramo no pueden entrar en la intersección. La cuarta regla representa que la celda conserva su estado en cualquier otro caso no contemplado en las condiciones anteriores.

Puede verse que no existe la posibilidad de deadlock con esta definición de cruces. El *deadlock* aparece cuando existe un conjunto de vehículos donde todos están esperando que se vacíe la posición siguiente dentro del cruce que contiene otro vehículo del mismo conjunto. Una forma de deadlock sería que *todas* las celdas del anillo o cruce contengan vehículos y así se forma un ciclo, donde cada uno espera que el otro avance. Pero esto no puede ocurrir pues si se tiene un cruce de *k* celdas, que contiene k-1 vehículos, el k_ésimo auto nunca puede ingresar porque las reglas del comportamiento de los tramos piden que por lo menos haya en el cruce 2 celdas vacías (celda de ingreso y la de atrás). Por lo tanto, en el cruce nunca puede haber más k-1 autos y entonces siempre se tiene por lo menos una celda vacía. Por consiguiente, siempre existe algún auto que no está esperando que se vacíe una celda. De esta forma queda demostrado que no puede haber ciclos de espera *dentro del cruce*.

Otra variante de deadlock puede darse cuando un auto espera para salir por una celda particular hasta que ésta se desocupe y genera un ciclo de esperas. En este caso, para evitar que ello ocurra al llegar a una salida sólo se intenta salir chequeando la función aleatoria en caso de tener lugar para hacer el movimiento, caso contrario permanece girando. Si le toca salir y la celda está ocupada entonces sigue girando. El comportamiento así descripto fue modelado en las reglas de los cruces.

Otro problema que podría presentarse es el de *inanición*, es decir que un auto nunca pueda salir del cruce y permanezca girando. Esto debería ser considerado para que estos vehículos que permanecen durante demasiado tiempo sin poder salir, tengan una mayor chance de abandonar el cruce al pasar por cada salida.

1.3.3. Restricciones y generalidades del Lenguaje

En esta sección se definen restricciones del lenguaje relacionadas tanto con los cruces como con los tramos; que luego permitirán su correcto acoplamiento.

Para poder establecer el orden en que los diversos tramos se acoplan a un mismo cruce, es decir cuáles son las celdas de cada modelo que se comunican entre sí; se utiliza la función Ports_In_Out definida en el Apéndice A. Ella además permite establecer cuáles son las celdas de entrada y cuáles las de salida para cada cruce, partiendo de la especificación de los tramos que lo tienen como extremo y chequeando el sentido de circulación de los vehículos en ellos. Así se establece un ordenamiento de los tramos que se acoplan al mismo cruce, de forma tal que los tramos más cercanos entre sí ocupen posiciones vecinas, esto se logra ordenándolos según el ángulo de inclinación respecto a la recta y = y1, donde las coordenadas del cruce son (x1,y1).

La función Ports_In_Out tiene como precondición que no existan más de 2 tramos con igual ángulo de inclinación acoplados al mismo cruce; y si hay 2 entonces deben tener diferente dirección. Esto permite un ordenamiento determinístico de los tramos acoplados a cada cruce. Para garantizar el cumplimiento de la precondición se debe establecer una restricción para la definición de los tramos. Para cada t $\in$ Tramos, con t = (p1, p2, n, a, dir, max), se construyen los conjuntos:

T_{p1} = { t' / t' $\in$ Tramos $\wedge$ t' = (p1', p2', n', a', dir', max') $\wedge$ t $\neq$ t' $\wedge$ [(p1 = p1' $\wedge$ Inclinación(p1,p2) = Inclinación(p1',p2')) $\vee$ (p1 = p2' $\wedge$ Inclinación(p1,p2) = Inclinación(p2',p1'))] }

T_{p2} = { t' / t' ∈ Tramos ∧ t' = (p1', p2', n', a', dir', max') ∧ t ≠ t' ∧ [(p2 = p1' ∧ Inclinación(p2,p1) = Inclinación(p1',p2')) ∨ (p2 = p2' ∧ Inclinación(p2,p1) = Inclinación(p2',p1'))] }

Donde, Inclinación es una función que devuelve el ángulo entre el segmento determinado por los puntos que recibe como parámetro y la recta y = y1, aquí y1 es el desplazamiento del primer parámetro sobre el eje y. Esta función se encuentra definida en el Apéndice A. El conjunto T_{p1} contiene todos los tramos con uno de sus extremos en p1 de igual inclinación que t, pero t ∉ T_{p1}. De forma análoga, el conjunto T_{p2} contiene todos los tramos con uno de sus extremos en p2 de igual inclinación que t, pero t ∉ T_{p2}. Sobre estos conjuntos se imponen las siguientes condiciones:

$$\#(T_{p1}) \leq 1 \wedge \#(T_{p2}) \leq 1 \wedge$$
$$\left[\begin{array}{c}\left(\#(T_{p1}) = 1 \wedge t' \in T_{p1} \wedge t' = (p1', p2', n', a', dir', max')\right) \\ \Downarrow \\ \left((p1 = p1' \wedge dir \neq dir') \vee (p1 = p2' \wedge dir = dir')\right)\end{array}\right] \wedge$$
$$\left[\begin{array}{c}\left(\#(T_{p2}) = 1 \wedge t' \in T_{p2} \wedge t' = (p1', p2', n', a', dir', max')\right) \\ \Downarrow \\ \left((p2 = p1' \wedge dir = dir') \vee (p2 = p2' \wedge dir \neq dir')\right)\end{array}\right]$$

De esta forma, se garantiza que cada cruce se acopla con a lo sumo 2 tramos de igual inclinación, pero deben tener distinta dirección.

Observaciones:

1) Cabe destacar que los tramos que se intersectan en puntos que no son sus extremos, no se definen como cruces y en el modelo no serán considerados como calles comunicadas donde los vehículos pueden pasar de una a la otra. Esto podría ser pensado como que se modela un puente, túnel o autopista donde un tramo cruza por encima del otro.
2) Las manzanas se pueden obtener a partir del conjunto de Tramos pues son los sectores bordeados por ellos.

1.3.4. Acoplamiento entre Cruces y Tramos

El acoplamiento de estos modelos representa el movimiento de vehículos que ingresan al (salen del) cruce desde (hacia) algún tramo. Un cruce c = (p, maxc) tiene influencias sobre los tramos t que lo tienen como un extremo, es decir, el conjunto de modelos que influye el Cell-DEVS de c es el siguiente:

I_c = { M_t / t ∈ Tramos ∧ t = (p1, p2, n, a, dir, max) ∧ (p1 = p OR p2 = p) }

Notación 4

M_i representa al modelo DEVS o Cell-DEVS definido para i, donde i es un elemento del lenguaje de especificación (por ejemplo, i puede ser un tramo, cruce, semáforo, etc.).

Luego un tramo t tiene definidas las influencias sobre los modelos de los dos cruces que tienen como extremo, es decir

$$I_t = \{M_{c1}\} \cup \{M_{c2}\}, \text{ si } t = (p1, p2, n, a, dir, max) \text{ y } (\exists\ v1, v2 \in N : c1, c2 \in Cruces \wedge c1 = (p1, v1) \wedge c2 = (p2, v2))$$

Para completar la definición del acoplamiento entre cruces y tramos hay que establecer a través de qué ports se comunican. Al definir el modelo Cell-DEVS para un cruce, se agrega una celda por cada carril de cada tramo que lo tiene como extremo; a través de ella se producirá el ingreso (la salida) del vehículo al (del) cruce. Por lo tanto, cada celda del borde (columnas 0 y k-1, si el tramo tiene k celdas) de cada tramo tiene una celda del cruce asociada, con la que debe intercambiar la información sobre su estado. Así cada celda del cruce presenta el port $Y_{\eta+1}$ para que la celda del tramo con la que se comunica reciba su estado por el port $X_{\eta+1}$. Luego, cada celda del borde del tramo presenta el port $Y_{\eta+1}$ para que la celda del cruce asociada reciba su estado por el port $X_{\eta+1}$. Estos ports interconectan la celda del cruce con la del tramo, para modelar el avance derecho de los vehículos de un modelo a otro, de acuerdo a la dirección de circulación definida en los tramos.

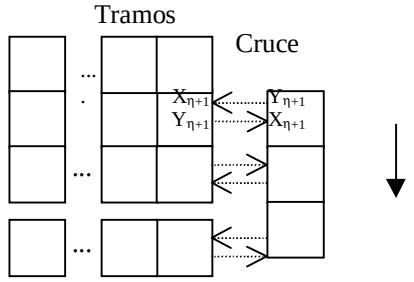


Figura 29 - Ports de acoplamiento entre tramos y cruces

Los ports se acoplan definiendo la función Z. Para establecer el índice de celda del cruce y la del tramo que tiene la conexión se utiliza la función Ports_In_Out (definida en el apéndice A) que realiza dicha tarea y devuelve dos conjuntos, el primero contiene cada tramo de ingreso al cruce que está conectado al mismo con el índice de la primera celda del cruce con la que se acopla; y el otro contiene la misma información pero para los tramos de salida. Esta función establece un ordenamiento de los tramos que se acoplan al mismo cruce, de forma tal que los tramos más cercanos entre sí ocupen posiciones vecinas dentro del mismo, esto se logra ordenándolos según el ángulo de inclinación respecto a la recta y = y1 con p = (x1, y1). Utilizando Ports_In_Out se obtienen los conjuntos I y O de la siguiente manera:

$$\{I, O\} = Ports\_In\_Out(c, Tramos)$$

Para cada $(t,i) \in I$ con t = (p1, p2, n, a, dir, max) se debe conocer la cantidad de celdas del tramo porque el acoplamiento se realiza en la primera (0) y última celda de cada carril (k-1, si k es la cantidad de celdas del tramo t). Esto se obtiene calculando la longitud del tramo (a partir de p1 y p2), y luego dividiéndola por el tamaño definido para la celda (en el Apéndice A se encuentra definida la función Ctd_Celdas). De esta forma la cantidad de celdas que tendrá t se calcula como:

$$k = Ctd\_Celdas(t)$$

La definición de Z es:

$$Z_{tc} : Y_{\eta+1}(j, k-1)_t \rightarrow X_{\eta+1}(0,i+j)_c, \forall\ (j \in N, j \in [0, n-1])$$
$$Z_{ct} : Y_{\eta+1}(0,i+j)_c \rightarrow X_{\eta+1}(j, k-1)_t, \forall\ (j \in N, j \in [0, n-1])$$

Para cada $(t,i) \in O$ con t = (p1, p2, n, a, dir, max)

$$Z_{ct} : Y_{\eta+1}(0,j+i)_c \rightarrow X_{\eta+1}(n\text{-}1\text{-}j, 0)_t, \forall (j \in N, j \in [0, n\text{-}1])$$
$$Z_{tc} : Y_{\eta+1}(n\text{-}1\text{-}j, 0)_t \rightarrow X_{\eta+1}(0,j+i)_c, \forall (j \in N, j \in [0, n\text{-}1])$$

1.4. Elementos de Control

En esta sección se incorporan construcciones del lenguaje para representar otras características que condicionan el movimiento de vehículos, como ser esquinas con semáforos, barreras de trenes, obras que cortan parte de la calle, baches, etc. Para cada uno de ellos primero se describe su especificación y luego se modela usando los formalismos DEVS y Cell_DEVS.

1.4.1. Semáforos

Los semáforos se especifican como:

$$CrucesSemáforos = \{ c / c \in Cruces \}$$

Cada cruce de este conjunto representa una esquina con semáforos. Es decir, los vehículos que llegan a la intersección deben chequear el color del semáforo para determinar si pueden avanzar.

La presencia de semáforos se representa utilizando modelos adicionales al cruce y tramos afectados. Se define un modelo DEVS (Semáforo) para cada calle de la intersección, que informa del color del semáforo a las celdas del tramo. Luego, por cada cruce se construye un modelo DEVS (Sincronizador) encargado de avisar a cada semáforo cuándo le corresponde la luz verde, es decir sincroniza todos los semáforos del cruce. Estos modelos se grafican en la Figura 30.

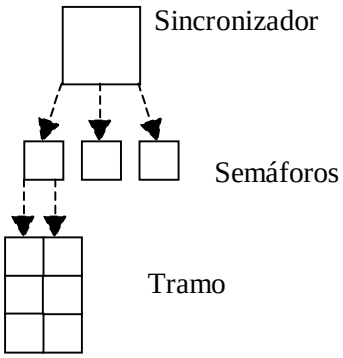


Figura 30 – Modelos para representar semáforos

A continuación se describen 2 modelos posibles para le sincronizador, el primero siempre está en funcionamiento durante toda la simulación; mientras que el otro deja de funcionar por períodos. Más adelante se presenta el modelo para los semáforos y las reglas para los autos dentro de los tramos que deben chequear su color antes de avanzar.

Para cada (c, maxc) $\in$ CrucesSemáforos, se define un modelo para el sincronizador de semáforos como:

$$Sincro1(\#sem) = < I, X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D >$$

El parámetro #sem indica la cantidad de semáforos que serán regulados por este sincronizador (la cantidad de semáforos equivale a la cantidad de tramos de ingreso del cruce que regula) y se obtiene a partir del conjunto de tramos de ingreso al cruce (c, maxc) que se construye como:

$$T_{in} = \{ t / t \in \text{Tramos} \wedge t = (c1, c2, n, a, dir, max) \wedge [(c1 = c \wedge dir = 0) \vee (c2 = c \wedge dir = 1)] \} \quad (a)$$

El parámetro buscado es la cantidad de elementos T_{in}, es decir,

$$\#sem = \#(T_{in})$$

I = { ($y\text{-se-luz}_i$, binario) / $0 \leq i < \#sem$ }

X = ∅

Y = {0, 1}
/* los estados representan el rojo (1) y el verde (0) */

S :
Variables Descriptivas
SemVerde (entero positivo, $0 \leq semverde < \#sem$)
SemVerde indica cuál es el semáforo que obtendrá la luz verde en el próximo cambio.
Inicialización: SemVerde = 0, phase = activa.

Parámetro
t_{verde} (real o entero positivo), indica el tiempo durante el cual el semáforo permanece en verde.

$\delta_{ext}(s,e,x)$
/* No hace nada, no hay ports de entrada */

$\lambda(s)$
{
send 0 *to* $y\text{-se-luz}_{SemVerde}$ /* envía luz verde al semáforo correspondiente*/
send 1 (rojo) *to* $y\text{-se-luz}_j$ ($\forall j \ \ 0 \leq j < \#sem \wedge j \neq SemVerde$) /* envía luz roja al */
/* resto de los semáforos */
}

$\delta_{int}(s,e)$
{
case phase

activa:
SemVerde = (SemVerde + 1) mod #sem
phase = activa
$\sigma = t_{verde}$
pasiva:
/* Nunca ocurre pues no hay evento planificado */
end case
}

Este sincronizador funciona siempre y regula los semáforos de ingreso al cruce (c, maxc). El comportamiento modelado a través de estas funciones corresponde a enviar la luz verde a los semáforos conectados a este sincronizador en forma alternada. Cada semáforo recibirá la luz verde durante el tiempo t_{verde}, luego del cual volverá a rojo hasta que le vuelva a tocar su turno. En cada momento un único semáforo de todos los conectados al sincronizador recibe la luz verde; mientras tanto los demás están en rojo.

Otro modelo para representar el sincronizador podría además modelar que los semáforos con baja probabilidad dejan de funcionar. Para ello se utiliza una función aleatoria que hace que se detenga el funcionamiento por un cierto período. Este modelo se define como:

$$\text{Sincro2(\#sem)} = < I, X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D >$$

El parámetro #sem indica la cantidad de semáforos que serán regulados por este sincronizador y se obtiene de la misma forma que para el modelo Sincro1.

I = { (y-se-luz$_i$, binario) / $0 \leq i < \#sem$ }

$X = \varnothing$

$Y = \{0, 1\}$
/* los estados representan el rojo (1) y el verde (0) */

S :
Variables Descriptivas
SemVerde (entero positivo, $0 \leq$ semverde $<$ #sem)
SemVerde indica cuál es el semáforo que obtendrá la luz verde en el próximo cambio.
Funciona? (número binario)
Funciona? indica si el sincronizador está funcionando o no.
Inicialización: SemVerde = 0, Funciona? = 1, phase = activa.

Parámetros
t_{verde} (entero o real positivo), indica el tiempo durante el cual el semáforo permanece en verde.
$t_{reparación}$ (entero o real positivo), indica el tiempo durante el cual el semáforo permanece fuera de servicio.

δ_{ext}(s,e,x)
/* No hace nada, no hay ports de entrada */

λ(s)
{
case Funciona?
1:
send 0 (verde) *to* y-se-luz$_{SemVerde}$
send 1 (rojo) *to* y-se-luz$_j$ ($\forall$j $0 \leq j < \#sem \wedge j \neq SemVerde$)
0: /* No funciona */
send 0 *to* y-se-luz$_j$ ($\forall$j $0 \leq j < \#sem$)
end case
}

δ_{int}(s,e)
{
case phase
activa:
case Funciona?
1:
Funciona? = rand()
case Funciona?
1:

```
                                    SemVerde = (SemVerde + 1) mod #sem
                                    phase = activa  /* planifica evento para cambiar */
                                    σ = t_verde          /* color semáforo*/
                                0:
                                    phase = activa /* planifica evento para apagar */
                                    σ = 0           /* el semáforo */
                            end case
                    0:
                            SemVerde = 0 /* planifica evento para re-inicio de */
                            Funciona? = 1 /* funcionamiento del semáforo */
                            phase = activa
                            σ = t_reparación
            end case

    pasiva:
            /* Nunca ocurre pues no hay evento planificado */
end case
}
```

Cuando el sincronizador no funciona envía la señal de verde, pues esto hace que los autos de las calles no esperen que el semáforo cambie para ingresar al cruce. La presencia de semáforos sólo modifica el comportamiento de los vehículos prohibiendo el acceso al cruce mientras esté en rojo, por lo tanto enviar verde a todos permite modelar que el sincronizador no está funcionando.

El comportamiento modelado a través de estas funciones corresponde a enviar la luz verde a los semáforos conectados a este sincronizador en forma alternada; y con cierta probabilidad deja de funcionar y envía luz verde a *todos*.

La función rand() devuelve el valor 1 con alta probabilidad y 0 en otro caso. El significado del 1 es que el sincronizador está funcionando correctamente. Esta función debería tener una distribución que represente la frecuencia con que los semáforos de una esquina se descomponen.

Cada sincronizador es el encargado de regular o establecer qué luz debe encender cada semáforo de una esquina, mientras que cada semáforo tiene como función informar su color a cada carril de algún tramo de ingreso a este cruce. Así, por cada tramo t de ingreso al cruce (c, maxc), donde (c, maxc) $\in$ CrucesSemáforos y t $\in$ T_{in} (definido como en (a)); se define el siguiente modelo para representar el semáforo:

$$Sem(\#c) = < I, X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D >$$

El parámetro #c indica la cantidad de carriles que serán regulados por el semáforo (#c equivale a la cantidad de carriles del tramo) y #c = n, donde t = (c1, c2, n, a, dir, max).

I = { (y-t-luz_i, binario) / $0 \leq i < \#c$ } U { (x-si-luz, binario) }

X = {0 , 1}
/* los estados representan el rojo (1) y el verde (0) */

Y = {0, 1}
/* los estados representan el rojo (1) y el verde (0) */

S :

Variables Descriptivas
ColorSem(número binario)
ColorSem indica cuál es el color actual del semáforo.
Inicialización: ColorSem = 1, phase = activa.

δ_{ext}(s,e,x)
{
cuando recibe x en el port sincro
case phase
activa:
/* no debería ocurrir */
pasiva:
ColorSem = x
phase = activa
$\sigma = 0$
end case
}

λ(s)
{
send ColorSem *to* y-t-luz$_i$ ($\forall i \in N, 0 \leq i < \#c$)
}

δ_{int}(s,e)
{
case phase
activa:
phase = pasiva
$\sigma = \infty$
pasiva:
/* Nunca ocurre */
end case
}

Sem es un semáforo que regulará el ingreso de los vehículos de un *sólo tramo* al cruce. El comportamiento modelado por el semáforo consiste en recibir la luz (roja o verde) que le envía el sincronizador e informar la misma a las celdas del tramo sobre las que tiene influencia. Cuando el semáforo recibe el verde del sincronizador, habilita a los vehículos del tramo para que ingresen al cruce. Las celdas del cruce no necesitan conocer el estado del semáforo, pues con el estado de las celdas del tramo les alcanza para determinar si el auto avanza o no.

Para cada tramo t de ingreso al cruce (c, maxc), donde (c, maxc) $\in$ CrucesSemáforos y t $\in$ T_{in} (definido como en (a)); se debe reflejar en sus reglas de comportamiento la presencia del semáforo. Para estos tramos se modifica el comportamiento de las celdas de la última columna del espacio, que ahora tendrán un port externo adicional que indicará el color de la luz. Por lo tanto, del modelo definido para t sólo cambia la interface externa y la función de transición local de estas celdas.

Sea
M_t = < Xlist, Ylist, I, *X*, *Y*, n, t_1, t_2, η, N, C, B, Z, select > el modelo del tramo t.

En la interface del modelo M_t se agregan los ports:

$\{<X_{\eta+2}(i, t_2\text{-}1), \text{binario}> / \ 0 \leq i < t_1 \}$

Estos ports serán denotados de la siguiente forma:

Port	Nombre	Comentario
$X_{\eta+2}$	x-se-luz	Este port informa el color del semáforo a las celdas del tramo. Si su estado es 1 representa el rojo, si es 0 está en verde.

Además se modifica el comportamiento de las celdas borde de la columna t_2-1 del espacio, pues deben chequear la luz del semáforo. Ellas son:

$$\{(i, t_2\text{-}1) / \ 0 \leq i < t_1 \}$$

Todas estas celdas, sin importar la cantidad de carriles del tramo, tienen definida la regla Sale_Hacia_Cruce de la siguiente forma:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports	Delay
0	((0,0) = 1 and portvalue(x-c-haylugar) = 0)	send(1, y-c-hayauto)	inercial

Esta regla es la única que cambia por la introducción de semáforos en el cruce. Ahora para ingresar al cruce también se debe chequear que el semáforo esté en verde, esto se modela como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports	Delay
0	((0,0) = 1 and portvalue(x-c-haylugar) = 0 and portvalue(x-se-luz) = 0)	send(1, y-c-hayauto)	inercial

Esta nueva regla representa que un vehículo ingresa al cruce desde el tramo, si hay lugar y si el semáforo está en verde (portvalue(x-se-luz) = 0). Estas celdas siguen teniendo demora inercial para el movimiento de ingreso al cruce. Ahora, esta demora refleja que mientras un vehículo intenta acceder al cruce el semáforo se debe mantener en verde y las posiciones del cruce vacías. Si esto no ocurre, el auto espera hasta que se den todas las condiciones durante el período correspondiente a la demora inercial. Estas celdas envían 1 por el port de salida (hacia el cruce) si existe un auto en la celda que está habilitado (las celdas del cruce permanecieron vacías y el semáforo en verde durante la demora inercial correspondiente) para avanzar al cruce, caso contrario envían 0.

El resto de las reglas definidas para estas celdas no es alterado, pues representan el avance de vehículos hacia ellas y los vehículos no deben cruzar el semáforo.

Para cada cruce (c, maxc) $\in$ CrucesSemáforos se construye un modelo Sincro (M_{sin}) y además se crea un modelo Sem (M_{sem}) por cada tramo t de ingreso a (c, maxc).

Para establecer las influencias y acoplamientos se necesita el conjunto de tramos de ingreso a (c, maxc) y el orden con que se acoplan al cruce, que se obtienen utilizando la función Ports_In_Out. Ésta se muestra en el Apéndice A y devuelve dos conjuntos, el primero contiene cada tramo de ingreso al cruce que está conectado al mismo con el índice de la primera celda del cruce con la que se acopla; y el otro contiene la misma información pero para los tramos de salida. Esta función establece un ordenamiento de los tramos que se acoplan al mismo cruce, de forma tal que los tramos más cercanos entre sí ocupen posiciones vecinas, esto se logra

ordenándolos según el ángulo de inclinación respecto a la recta y = y1 con c = (x1,y1). A partir de los conjuntos que devuelve se puede obtener el parámetro buscado.

$$\{In, Out\} = Ports\_In\_Out((c, maxc), Tramos)$$

A partir del conjunto In se sabe cuáles son los tramos de ingreso a (c, maxc) y el orden en que los mismos reciben la luz del semáforo. Es decir, cada tupla $(t,j) \in In$, indica que el tramo t ingresa al cruce utilizando como primera celda de acoplamiento la j-ésima del cruce. De acuerdo al acoplamiento de estos tramos con el cruce (numeración de las celdas de cruce creciente) se establece el orden en que recibirán la luz verde.

El sincronizador influye sobre el comportamiento de los modelos Sem correspondientes a los tramos de ingreso al cruce (c, maxc). Esto es:

$$I_{sin} = \{ M_{sem_i} / 0 \leq i < \#(In) \}$$

Cada semáforo influye sobre el comportamiento del modelo del tramo de ingreso al cruce (c, maxc), según el orden establecido:

$$I_{sem_i} = \{ M_t / (t,j) \in In \wedge i = \#(\{(t',j') \in In / j' < j \})\}$$

Luego el acoplamiento se define como:

$Z_{sin\ sem_i} : (y\text{-}se\text{-}luz_i)_{sin} \rightarrow (x\text{-}si\text{-}luz)_{sem_i}, \forall (i \in N,\ 0 \leq i < \#(In))$

$Z_{sem_i\ t} : (y\text{-}t\text{-}luz_h)_{sem_i} \rightarrow X_{\eta+2}(h, k\text{-}1)_t,$

$\forall (h \in N, h \in [0, n\text{-}1]) \wedge (t,j) \in In \wedge i = \#(\{(t',j') \in In / j' < j \})$ Donde n es la cantidad de carriles de t ($n = t_1$ del modelo M_t) y k es la cantidad de celdas de cada carril ($k = t_2$ del modelo M_t).

1.4.2. Trenes

La influencia de los trenes sobre el flujo de vehículos es a través de los pasos a nivel (con y sin barreras) que impiden el avance de los autos por un determinado tiempo. Aquí se plantean 2 formas para especificar la presencia de trenes, en la sección 1.4.2.1 se definen como un trazado de vías, que es una secuencia de pasos a nivel sobre los tramos. El tren va siguiendo la secuencia definida, y su movimiento se modela con un espacio Cell-DEVS de una dimensión con demora de transporte. Mientras que en la sección 1.4.2.2, cada paso a nivel se define por separado, sin establecer un ordenamiento, sólo indicando sobre qué tramos se encuentran. En este caso, para cada paso a nivel se utiliza un modelo DEVS sin acoplamiento con los otros, que con cierta frecuencia impide que los vehículos avancen.

1.4.2.1. Definición de Trenes usando Cell-DEVS

El trazado de las vías del tren se especifica como:

$$RedDeVías = \{ Vías / Vías = \{ (t, \ell, seq) / t \in Tramos \wedge \ell \in N \wedge seq \in N \} \}$$

Cada elemento $Vías \in RedDeVías$ representa el trazado de las vías de algún ramal de trenes. Para especificarlo se indican los lugares donde se ubican los pasos a nivel (intersección entre las vías y las calles donde se permite el cruce de los vehículos). Cada tupla, $pn = (t, \ell, seq)$,

identifica la ubicación de un paso a nivel, es decir el tramo (t) y la distancia entre el comienzo del tramo y las vías (ℓ). Además indica el orden que le corresponde al paso a nivel (seq) para poder establecer la secuencia de avance del tren (en qué orden avanza sobre los pasos a nivel).

En cada paso a nivel puede haber o no una barrera que regule el flujo de vehículos. Ambos casos son modelados sin distinción puesto que los vehículos chequean por la presencia del tren que puede estar dada por las barreras bajas o porque perciben su cercanía.

Para simplificar el modelo, no se permite que las celdas de los tramos que se acoplan con los cruces (algunas además con semáforos) puedan contener pasos a nivel, esto se restringe de la siguiente forma:

$\forall Vías \in RedDeVías$:

$\forall((c1, c2, n, a, dir, max), \ell, seq) \in Vías$:

$long\_celda < \ell < Long\_\mathrm{Re}cta(c1, c2) - 2 * long\_celda$

Además los pasos a nivel para un trazado de vías particular forman una secuencia que comienza en 0 y se extiende hasta la cantidad de tuplas del conjunto.

$\forall Vías \in RedDeVías$:

$(\forall j \in N : 0 \leq j \leq \#(Vías) \Rightarrow \exists(t, \ell, j) \in Vías)$

Para cada elemento del conjunto RedDeVías se definen 2 modelos, el primero es un DEVS que representa la estación de donde parten los trenes y el segundo es un Cell-DEVS de una dimensión donde cada celda controla un paso a nivel de la vía. La estación lo único que hace es simular la partida de trenes cada cierto tiempo, de acuerdo a su frecuencia. El Cell-DEVS VíasTren representa el movimiento del tren que avanza con la velocidad modelada con demora de transporte. Estos modelos se grafican en la Figura 31.

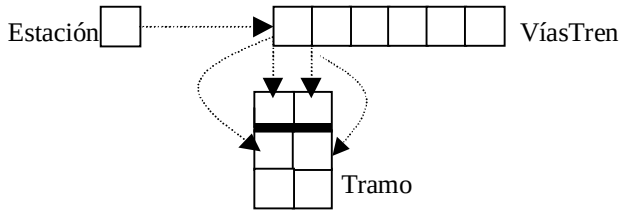


Figura 31 – Modelos para representar los trenes

Para cada Vías ∈ RedDeVías, se define su estación asociada como:

Estación = < I, X, S, Y, δ_{int}, δ_{ext}, λ, D >

I = { (y-vt-tren, binario) }

X = ∅

Y = {0, 1}

S :
Variables Descriptivas
tren_partiendo (número binario)
Inicialización: tren_partiendo = 1, phase = activa

Parámetros
frec (real positivo)
ξ(real positivo pequeño)

δ_{ext}(s,e,x)
/* No hace nada pues no hay ports de entrada */

λ(s)
{
send tren_partiendo *to* y-vt-tren /* informa de la partida del tren al modelo VíasTren */
}

δ_{int}(s,e)
{
case tren_partiendo
1:
tren_partiendo = 0 /* planifica evento para avisar que la estación */
phase = activa /* está vacía */
σ = ξ
0:
tren_partiendo = 1 /* planifica evento para el próximo tren */
phase = activa
σ = frec
end case
}

Este modelo representa la partida de trenes de acuerdo su frecuencia. La estación tiene un único port de salida (y-vt-tren) que lo acopla con un modelo VíasTren. Cuando su estado es 1 indica que está partiendo un tren desde la estación, por lo que comienza a circular por las vías correspondientes. Cuando su estado es 0, indica que no hay ningún tren listo para salir de la estación. Cada cierto tiempo, dado por el parámetro frec, se modela la partida de un nuevo tren, planificando el evento correspondiente. La constante ξ modela el tiempo en que tarda el tren hasta salir de la estación, es decir una vez transcurrido ésta vuelve a estar vacía (tren_partiendo = 0).

Para cada Vías ∈ RedDeVías, se define un espacio Cell-DEVS de una dimensión con demora de transporte para modelar el movimiento del tren sobre las vías. Cada celda del modelo representa un punto donde las vías cruzan una calle y se especifican como:

$$C_{0i} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = <η, P^x, P^y>, donde

η = 2;
P^x = { (X_1, binario), (X_2, binario) };
P^y = { (Y_1, binario), (Y_2, binario) }.

X = ***Y*** = {0, 1};

S:

$$s = \begin{cases} 1 \text{ si está el tren;} \\ 0 \text{ sino.} \end{cases}$$

N = { (0,-1), (0,0) } ;

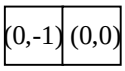


Figura 32 - Vecindario de la celda origen

delay = transport;

d = d_{tren} ($d_{tren} \in$ P);

λ, δ_{int} y δ_{ext} se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

τ: S x N → S se define de la siguiente manera,

Nuevo Estado	Estado del vecindario
0	(0,0) = 1
1	(0,-1) = 1
(0,0)	t /*En cualquier otro caso conserva su estado*/

El estado de una celda representa la presencia (s = 1) o ausencia (s = 0) del tren. Como sólo se permite que el tren avance hacia la posición de adelante, el vecindario necesario para establecer el nuevo estado de una celda está constituido por ella misma y la celda anterior.

Las demoras de transporte se utilizan para representar la velocidad del tren, es decir su llegada al próximo paso a nivel se demora el tiempo dado por la constante d_{tren}. Para simplificar el modelo se asume que el tiempo que le lleva al tren llegar de un paso a nivel al siguiente es el mismo; pues si no es así cada celda debería tener distinta demora que depende de la distancia real entre los puntos del plano que representan.

El movimiento de los trenes (especificado con la función τ) queda definido por 3 reglas. La primera representa que la celda se vacía luego que el tren haya pasado. La segunda representa el avance del tren desde el paso de nivel anterior hacia la celda de origen. Por último la tercer regla indica que el estado de la celda de origen no es modificado en cualquier otro caso. El comportamiento definido por estas reglas supone que el tren siempre avanza y que nunca encontrará otro adelante. De esta forma cada celda siempre recibe el tren que se encontraba en la celda anterior utilizando una demora de transporte fija para modelar el tiempo entre cada barrera. Estas simplificaciones son razonables teniendo en cuenta que el objetivo de incluir los trenes es ver cómo éstos afectan el comportamiento de los vehículos, y no lograr un modelado completo de su funcionamiento real que incluya además vías con bifurcaciones, retardos del servicio, descarrilamientos, etc.

El modelo acoplado correspondiente a las vías se define como:

VíasTren(k, Out) = < Xlist, Ylist, I, *X*, *Y*, n, {t_1,...,t_n}, η, N, C, B, Z, select >

donde,

Ylist = {(0,i) / i ∈ N ∧ 0 ≤ i < k }

Xlist = { (0,0) }

I = <P^x, P^y>

P^x = {<$X_{\eta+1}$(0,0), binario>}
P^y = {<$Y_{\eta+j}$(0,i), binario> / i ∈ N ∧ 0 ≤ i < k ∧ (i, #p) ∈ Out ∧ 1 ≤ j ≤ #p }

Estos ports serán denotados de la siguiente forma:

Port	Nombre	Comentario
$X_{\eta+1}$	x-e-tren	Este port se utiliza para recibir desde la estación la partida de un nuevo tren.
$Y_{\eta+j}$	y-t-tren	Estos ports se utilizan para informar a cada carril de cada tramo la presencia del tren en el paso a nivel. Existen 2 ports por cada carril del tramo acoplado a la celda del modelo.

X = { 0, 1 }

Y = { 0, 1 }

n = 1

$\mathbf{t_1}$ = k

η = 2

N = { (0,-1), (0,0) }

C = { C_{ij} / i = 0 ∧ j ∈ [0, k-1] }, donde cada C_{ij} es un componente Cell-DEVS con demora y la especificación de cada celda ha sido descripta antes de introducir el modelo acoplado.

B = { (0,0) }

Z se construye siguiendo la definición dada por el formalismo Cell_DEVS, instanciada con el vecindario de este espacio.

select = { (0,0), (0,-1) }

VíasTren(k, Out) modela un trazado de vías donde:

- k representa la cantidad de pasos a nivel que tienen las vías del tren y se obtiene contando los elementos de conjunto:

  Vías = { (t, ℓ, seq) / t ∈ Tramos ∧ ℓ ∈ N ∧ seq ∈ N }.

  Es decir, k = #(Vías); y

- el conjunto Out contiene la cantidad de ports para el acoplamiento con los tramos que tener cada celda, y se construye como:

  $$Out = \{ (i, 2*n) / i \in N \wedge 0 \leq i < k \wedge ((c1, c2, n, a, dir, max), \ell, i) \in Vías \}$$

  Cada celda tendrá tantos ports de salida externos como 2 veces la cantidad de carriles del tramo al que se acopla; pues la presencia del tren debe ser informada a 2 celdas de cada carril.

La especificación de este modelo representa el movimiento del tren, donde cada celda estará acoplada a un tramo e informará de la presencia del tren por las vías, prohibiendo la circulación de vehículos durante un lapso de tiempo.

El acoplamiento externo de este modelo se define a través de la celda (0,0) con una estación, es decir cuando el estado de este port externo de entrada es 1 indica el avance del tren hacia el modelo. Además cada celda tiene ports de salida externos hacia 2 celdas de cada carril del tramo donde se encuentra el paso a nivel correspondiente para informar la presencia del tren (s = 1). Uno de esos ports se utiliza para la celda atravesada por las vías, el otro port es para la celda que está a continuación de ellas. Estas últimas deben conocer si el tren pasó o no para saber si los vehículos se quedaron o no en la celda anterior.

El comportamiento para las celdas borde es distinto al definido anteriormente. Para la celda (0,0) se define el siguiente vecindario y comportamiento:

$\boldsymbol{\eta} = 1$

$\mathbf{N} = \{ (0,0) \}$

La función τ para esta celda se define como:

Nuevo Estado	Estado del vecindario
0	(0,0) = 1
1	portvalue(x-e-tren) = 1
(0,0)	t /*En cualquier otro caso conserva su estado*/

La celda (0,0) presenta comportamiento distinto porque recibe los trenes desde la estación, en vez de su vecino anterior. Por lo tanto las reglas son las mismas que las definidas para el resto de las celdas pero tienen un port de entrada que representa el estado de la estación.

El resto de los parámetros de esta celda no cambian.

Cada tupla (t, ℓ, seq) $\in$ Vías, donde Vías $\in$ RedDeVías; representa que las vías atraviesan el tramo t. De esta forma se deben definir los cambios sobre el modelo del tramo t que reflejan la presencia del paso a nivel. Para establecer cuáles son las celdas afectadas por la barrera basta dividir ℓ por la longitud de la celda, es decir

$$col = \lceil \ell / long\_celda \rceil$$

y se obtienen las celdas:

$$C_{pn} = \{ (i, col) / 0 \leq i \leq n-1 \} \cup \{ (i, col+1) / 0 \leq i \leq n-1 \}$$

donde n es la cantidad de carriles de t. Es decir, las celdas afectadas por el tren son aquellas que se encuentran justo antes y justo después de las vías, porque las primeras no avanzan si el tren está cerca y en ese caso las otras no reciben autos. Además el único movimiento permitido para los autos que atraviesan las vías es recto. Estas celdas se pueden ver en la Figura 33.

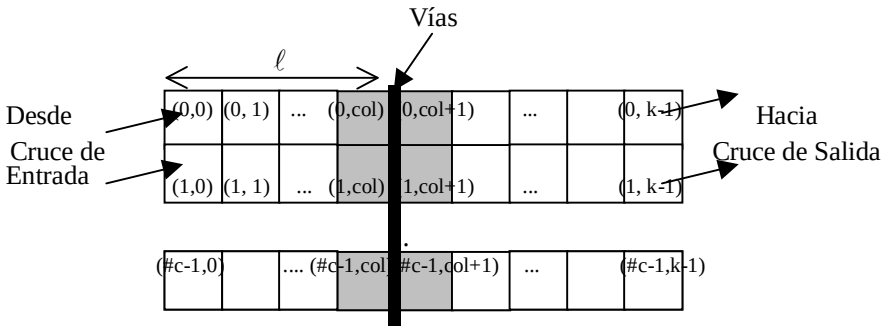


Figura 33 - Tramo con paso a nivel

Del modelo especificado para el tramo t (M_t), sólo se modifica el comportamiento de las celdas de las columnas col y col+1 (celdas del conjunto C_{pn}), que ahora tendrán un port externo adicional que indicará la presencia o ausencia del tren. Por lo tanto, cambia la interface externa y la función de transición local de esas celdas.

Sea

M_t = < Xlist, Ylist, I, *X*, *Y*, n, t_1, t_2, η, N, C, B, Z, select > el modelo del tramo t.

En la interface del modelo M_t se agregan los ports:

$$\{<X_{\eta+3}(i,col), binario> / \ 0 \le i < t_1 \} \ U \ \{<X_{\eta+3}(i,col+1), binario> / \ 0 \le i < t_1 \}$$

y serán denotados como:

Port	Nombre	Comentario
$X_{\eta+3}$	x-vt-tren	Este port informa la presencia del tren en el paso a nivel. Si su estado es 1 representa que está pasando el tren, si es 0 las vías están libres.

Las celdas del conjunto C_{pn} deben chequear que no pase el tren para poder cruzar las vías. Ellas son:

$$\{ (i, col) / 0 \le i < t_1 \} \ U \ \{ (i, col+1) / 0 \le i < t_1 \}$$

Para las celdas del conjunto $\{ (i, col) / 0 \le i < t_1 \}$ se modifican las reglas que definen el comportamiento, sin importar la cantidad de carriles del tramo, de la siguiente forma. Se eliminan las reglas Sale_Hacia_CarrilIz, Sale_Hacia_CarrilDer_SinPrioridad y Sale_Hacia_CarrilDer_ConPrioridad, para las celdas que las tengan definidas. Esto se hace para que el movimiento al cruzar las vías sea en línea recta, impidiendo adelantamientos. Por otro lado, la regla Sale_Hacia_Adelante definida como:

Nuevo Estado	Estado del vecindario	Delay	d
0	(0,0) = 1 and (0,1) = 0	transport	Conversión _Demora(velocidad(max))

es cambiada por:

Nuevo Estado	Estado del vecindario	Delay	d
0	((0,0) = 1 and (0,1) = 0 and portvalue(x-vt-tren) = 0)	inercial	$d_{vías}$

Donde **d** = $d_{vías}$ ($d_{vías} \in$ P), es una demora constante mayor que la demora que representa la velocidad normal de los autos. Esta regla representa que para poder abandonar la celda origen es necesario chequear que el tren no esté pasando (portvalue(x-vt-tren) = 0). Otro cambio que se introduce es la utilización de demora inercial fija ($d_{vías}$) más grande que la demora de las demás celdas para representar que un auto circula más lento cuando atraviesa las vías. Además esta demora permite modelar que los vehículos al llegar al paso a nivel, avanzan sólo si durante un cierto tiempo (demora inercial) tienen el camino libre (vías sin tren), caso contrario esperan a que pase el tren (desalojo de estado).

Por último, el resto de las reglas que son aquellas que representan la llegada de vehículos a estas celdas no cambian (Llega_Desde_Atrás, Llega_Desde_CarrilDer, Llega_Desde_CarrilIz_SinPrioridad y Llega_Desde_CarrilIz_ConPrioridad).

Por otro lado, las celdas que se encuentran a continuación de las vías también se ven afectadas por su presencia, pues para saber si un vehículo avanzará sobre ellas deben verificar que no haya un tren en las vías. Ellas son las celdas del conjunto { (i, col+1) / $0 \leq i < t_1$ }. Su comportamiento se modifica eliminando las reglas LLega_Desde_CarrilDer, LLega_Desde_CarrilIz_SinPrioridad y LLega_Desde_CarrilIz_ConPrioridad, para las celdas que las tengan definidas. Esto se hace para que el movimiento al cruzar las vías sea en línea recta, impidiendo adelantamientos. Por otro lado, la regla Llega_Desde_Atrás definida como:

Nuevo Estado	Estado del vecindario	Delay	d
1	(0,0) = 0 and (0,-1) = 1	transport	Conversión_ Demora(velocidad(max))

es cambiada por:

Nuevo Estado	Estado del vecindario	Delay	d
1	(0,0) = 0 and (0,-1) = 1 and portvalue(x-vt-tren) = 0	inercial	$d_{vías}$

Esta regla representa que para poder avanzar hacia la celda origen, el vehículo que se encuentra antes de las vías, debe verificar que no esté pasando el tren (portvalue(x-vt-tren) = 0). Otro cambio que se introduce es la demora inercial fija, pues para atravesar las vías éstas deben permanecer vacías durante el tiempo que les lleva a los autos cruzarlas.

Por último, las reglas que representan la partida de vehículos desde estas celdas no cambian, estas son Sale_Hacia_Adelante, Sale_Hacia_CarrilIz, Sale_Hacia_CarrilDer_ConPrioridad y Sale_Hacia_CarrilDer_SinPrioridad.

Para terminar, es necesario definir cómo se hace el acoplamiento entre Trenes y Tramos.

Aquí, cada elemento V del conjunto RedDeVías, tendrá la forma:

$$V = \{ (t, \ell, seq) / t \in Tramos \wedge \ell \in N \wedge seq \in N \}$$

Para V se construye un modelo Estación (M_E) y otro VíasTren (M_{VT}).
Cada tupla (t, ℓ, seq) $\in$ V genera las siguientes influencias:

/* Definición de las influencias $\{I_i\}$*/
$I_E = \{ M_{VT} \}$
$I_{VT} = \{ M_t \}$

Para establecer los acoplamientos se necesita obtener la columna del tramo t cruzada por las vías. Para ello basta dividir ℓ por la longitud de la celda, es decir

$$col = \lceil \ell / long\_celda \rceil$$

Además n, es la cantidad de carriles de t, que se obtiene de su especificación o del modelo M_t (ctd. de carriles es t_1). Con estos parámetros el acoplamiento de los modelos se define como:

$Z_{E,VT} : y\text{-}vt\text{-}tren_E \rightarrow X_{\eta+1}(0,0)_{VT}$
$Z_{VT,t} : Y_{\eta+j}(0,seq)_{VT} \rightarrow X_{\eta+3}(j\text{-}1,col)_t \ \forall j \ (j \in N \wedge 1 \leq j \leq n)$
$Z_{VT,t} : Y_{\eta+n+j}(0,seq)_{VT} \rightarrow X_{\eta+3}(j\text{-}1,col+1)_t \ \forall j \ (j \in N \wedge 1 \leq j \leq n)$

1.4.2.2. Definición de Trenes usando modelos atómicos DEVS

Las vías se especifican como:

$$Vías = \{ (t, \ell) / t \in Tramos \wedge \ell \in N \}$$

Cada tupla, pn = (t, ℓ), identifica la ubicación de un paso a nivel, es decir el tramo (t) y la distancia entre el comienzo del tramo y las vías (ℓ).

Para representar cada paso a nivel se define un modelo DEVS que indica a las celdas del tramo la presencia del tren. Por lo tanto para cada tupla se define uno de estos modelos.

Para cada, pn = ((c1, c2, n, a, dir, max), ℓ) se define su modelo asociado como:

$$PasoNivel = < I, X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D >$$

I = { ($y\text{-}t\text{-}tren_i$, binario) / $i \in N \wedge 0 < i \leq 2n$ }

$X = \varnothing$

$Y = \{0, 1\}$

S :
Variables Descriptivas
pasa_tren (número binario)
Inicialización: pasa_tren = 1, phase = activa

Parámetros
frec (real positivo)
tcruce(real positivo)

$\delta_{ext}(s,e,x)$

/* No hace nada pues no hay ports de entrada */

λ(s)
{
send pasa_tren *to* y-t-tren$_i$ $\forall(i \in N \wedge 0 < i \leq 2n)$
/* informa de la presencia del tren al modelo Tramo */
}

δ_{int}(s,e)
{
case pasa_tren
1:
pasa_tren = 0 /* planifica evento para avisar que terminó */
phase = activa /* de pasar el tren */
σ = tcruce
0:
pasa_tren = 1 /* planifica evento para el próximo tren */
phase = activa
σ = frec
end case
}

Este modelo representa la presencia del tren en el paso a nivel con frecuencia frec y las vías permanecen ocupadas durante el tiempo indicado por el parámetro tcruce. Cuando el tren se encuentra en el paso a nivel, se actualizan los ports de salida con el valor 1, para que las celdas de los tramos atravesadas por las vías no permitan el avance de vehículos. Las celdas que deben conocer el estado del paso a nivel son aquellas que se encuentran en las columnas anterior y posterior a las vías. Por lo tanto, son necesarios n ports para cada columna (2n ports en total), donde n es la cantidad de carriles.

Para que este modelo quede completo, es necesario que los diversos pasos a nivel estén sincronizados para que simulen la circulación de un tren. Pues sino, cada barrera baja en forma independiente de las demás y no se modela el avance de un tren de una estación a otra. Por lo tanto, se debe definir otro modelo que se encargue de informar a cada paso a nivel en qué momento debe bajar las barreras por primera vez; pues luego cada uno continúa con su funcionamiento normal de acuerdo a su frecuencia. Esto escapa a los alcances del presente trabajo, pero queda planteado para una futura extensión.

Cada tupla (t, ℓ) $\in$ Vías, representa que las vías atraviesan el tramo t. De esta forma se deben definir los cambios sobre el modelo de t que reflejan la presencia del paso a nivel. Estos cambios son los mismos que fueran planteados en la sección 1.4.2.1, para los tramos con trenes mapeados con modelos Cell-DEVS.

Cada tupla (t, ℓ) $\in$ Vías genera las siguientes influencias:

/* Definición de las influencias {I_i} */
I_{PN} = { M_t } donde PN representa al modelo del paso a nivel de la tupla (t,ℓ)

Para establecer los acoplamientos se necesita obtener la columna del tramo t cruzada por las vías. Para ello basta dividir ℓ por la longitud de la celda, es decir

$$col = \lceil \ell / long\_celda \rceil$$

Además n, es la cantidad de carriles de t, que se obtiene de su especificación o del modelo M_t (ctd. de carriles es t_1). Con estos parámetros el acoplamiento de los modelos se define como:

$Z_{PN,t}$: y-t-tren$_j$ $\rightarrow$ $X_{\eta+3}$(j-1,col)$_t$ $\forall$j (j $\in$ N $\wedge$ 1 $\leq$ j $\leq$ n)

$Z_{PN,t}$: y-t-tren$_{j+n}$ $\rightarrow$ $X_{\eta+3}$(j-1,col+1)$_t$ $\forall$j (j $\in$ N $\wedge$ 1 $\leq$ j $\leq$ n)

1.4.3. Obras

Las obras son secciones de calles deshabilitadas para la circulación de vehículos, debido a la presencia de obreros trabajando. Las obras se especifican como:

Obras = { (t, ni, ℓ, #n) / t $\in$ Tramos $\wedge$ t = (c1, c2, n, a, dir, max) $\wedge$ ni $\in$ [0, n-1] $\wedge$ ℓ $\in$ N $\wedge$ #n $\in$ [1, n+1-ni] $\wedge$ #n $\equiv$ 1 mod 2 }

Cada tupla del conjunto, o = (t, ni, ℓ, #n), identifica el tramo (t) donde se halla la obra, el primer carril (ni) afectado por la obra, la distancia sobre el carril ni que existe entre la columna central de la obra y el comienzo del tramo, y la cantidad de carriles que ocupa la obra (#n). Cada tupla especifica un rombo sobre un tramo por donde los vehículos no pueden circular. Por lo tanto sólo se pueden definir obras con esa forma, pero esto no implica una limitación en la expresividad del lenguaje porque en general son construidas así para permitir el desvío gradual del tráfico. Se pide que la cantidad de carriles sea impar para poder armar el rombo (si fuera par se obtiene un rombiode y no existe una celda central) y que el carril inicial (ni) más la cantidad de carriles que ocupa (#n) no sea más grande que la cantidad de carriles totales del tramo.

Las obras, (t, ni, ℓ, #n), deben estar completamente contenidas en el tramo. Como su altura se define en función de los carriles, se garantiza que no supere los bordes superiores e inferiores del tramo. Pero hay que pedir que ocurra lo mismo para el borde derecho e izquierdo. Esta restricción se escribe como:

$\forall((c1, c2, n, a, dir, max), ni, l, \#n) \in Obras$:

$$0 \leq l + (\#n - 1/2) * long\_celda \leq Long\_Recta(c1, c2)$$

donde Long_Recta es una función definida en el Apéndice A, que recibe un par de puntos del plano como parámetros y calcula la distancia que hay entre ellos; y long_celda es una constante que representa el largo de las celdas.

Para evitar complicar el modelo permitiendo que las celdas que se acoplan con los cruces puedan contener obras o ser las celdas adyacentes anteriores a la obra, esto se restringe de la siguiente forma:

$\forall((c1, c2, n, a, dir, max), ni, l, \#n) \in Obras$:

$$(2 * long\_celda < l - (\#n - 1/2) * long\_celda) \wedge$$
$$(l + (\#n - 1/2) * long\_celda < Long\_Recta(c1, c2) - long\_celda)$$

Tampoco se permite que dos ó más obras se solapen., entre cualquier par de obras debe existir por lo menos un carril o sector con comportamiento normal (sin obras). Esta restricción se puede representar como:

$$\forall o_1, o_2 \in Obras : o_1 \cap o_2 = \varnothing$$

En la Figura 34 se muestra gráficamente la especificación de una obra.

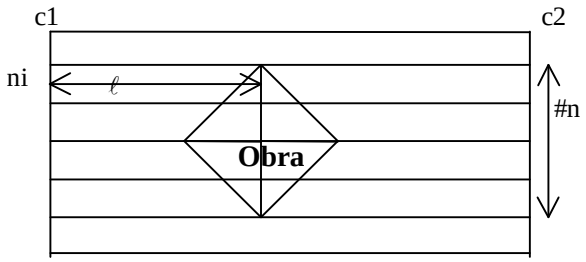


Figura 34 – Tramo con obra

Para representar una obra o = (t, ni, ℓ, #n) en el modelo Cell-DEVS correspondiente al tramo t, se define un comportamiento diferente para las celdas dentro del rombo y las que se encuentran adelante del mismo. Las primeras que representan a la obra, tienen estado constante en 0; pues en ellas no hay vehículos. Las otras deben hacer que los autos esquiven a las celdas en obras, restringiendo sus movimientos acordemente. Ambos tipos de celdas se ilustran en la Figura 35. Por otro lado, las celdas que se encuentran a continuación del rombo no modificarán su comportamiento pues desde el port de las celdas en obras recibirán siempre el estado vacío y nunca recibirán un vehículo desde ellas.

			O					
		O	O	X				
		O	X	X	X			
		O	O	X				
			O					

O representa celda anterior a la obra
X representa celda dentro de la obra

Figura 35 – Celdas de la obra y anteriores

Para que los vehículos no queden atascados entre las celdas de las obras, se ha elegido la forma de rombo, pero si algún extremo de éste toca el borde superior o inferior del tramo, se deben agregar algunas celdas más. En estos casos se completa la figura del rombo con un triángulo sobre el borde correspondiente. Por lo tanto se considera por separado el caso en que el rombo no toca los bordes del tramo, el caso en que toca el borde superior y por último cuando entra en contacto con el inferior.

En el primer caso, para representar la obra o = (t, ni, ℓ, #n), se considera que el rombo no toca bordes del tramo, es decir, ni ≠ 0 y ni + #n ≠ t_1 (t_1 es el parámetro del modelo M_t, definido para el tramo t, que acota la cantidad de filas del espacio). Aquí se debe determinar a qué celdas del tramo afecta, es decir aquellas que forman parte de la obra y las que se encuentran en las posiciones anteriores. La celda del centro de la obra se obtiene como:

$$(\text{fil, col}) = \left(ni + (\#n\text{-}1)/2, \lceil l / long\_celda \rceil\right)$$

El conjunto de celdas dentro de la obra se define como:

$$C_{Obras} = \{ (\text{fil} + i, \text{col} + j) / -(\#n-1)/2 \le j \le (\#n-1)/2 \ \wedge$$
$$-\left(\frac{\#n-1}{2}\right)+|j| \le i \le \left(\frac{\#n-1}{2}\right)-|j| \ \} = \{ (\text{fil} + i, \text{col} + j) / \ |j| \le (\#n-1)/2 \ \wedge$$
$$|i| \le \left(\frac{\#n-1}{2}\right)-|j| \ \}$$

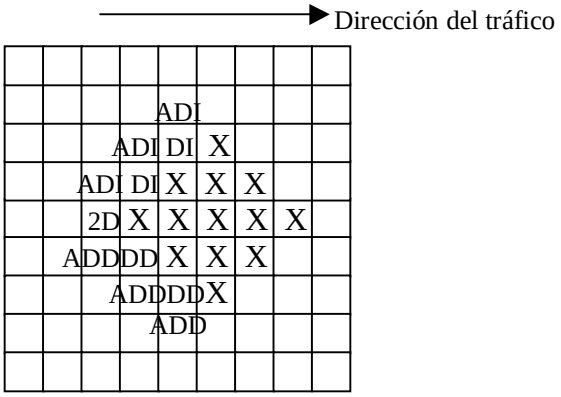
La función τ para estas celdas se define como:

Nuevo Estado	Estado del vecindario

0	t /* En cualquier caso tiene estado cero */

Las celdas de la obra no pueden contener vehículos, por tal motivo su estado es siempre 0.

Las celdas que se encuentran antes de la obra se pueden agrupar según los movimientos permitidos para los vehículos que se encuentran en ellas. Las celdas comunes tienen 3 tipos de movimientos, hacia adelante, hacia adelante en diagonal derecha y hacia adelante en diagonal izquierda. Las celdas anteriores a las obras tendrán sólo algunos de ellos de acuerdo a su ubicación. Por lo tanto habrá celdas que sólo pueden avanzar en alguna diagonal o en ambas diagonales o hacia adelante y en alguna diagonal. Estos tipos de celdas se muestran en la siguiente figura.



Donde X representa que la celda está dentro de la obra, DI representa que la celda sólo se puede mover hacia la Diagonal Izquierda, DD sólo en Diagonal Derecha, ADI indica que la celda tiene 2 movimientos posibles hacia Adelante y hacia la Diagonal Izquierda, ADD hacia Adelante y hacia la Diagonal Derecha y 2D hacia las 2 Diagonales.

Figura 36– Celdas anteriores a las obras

Así las celdas anteriores a la obra son de los tipos 2D (2 diagonales), DD (diagonal derecha), DI (diagonal izquierda), ADD (hacia adelante y diagonal derecha) y ADI (hacia adelante y diagonal derecha).

El conjunto de celdas con movimiento en dos diagonales se define como:

Celda_2D = { (fil, col - 1 - (#n-1)/2) }

La función τ para estas celdas se obtiene modificando el comportamiento normal definido para los vehículos en los tramos. Desde estas celdas los autos no pueden avanzar derecho hacia adelante (el vecino (0,1) está dentro de la obra). Por lo tanto, se elimina la regla Sale_Hacia_Adelante. Luego, el resto de las reglas no cambian.

El conjunto de celdas con movimiento diagonal izquierda se define como:

$$\text{Celdas\_DI} = \{ (fil + i, col + j) / -(\#n-1)/2 \le j \le -1 \wedge i = -\left(\frac{\#n-1}{2}\right) + |j| - 1 \ \}$$

La función τ para estas celdas se obtiene modificando el comportamiento normal definido para los vehículos en los tramos. Desde estas celdas los autos sólo pueden avanzar usando la diagonal izquierda (hacia el vecino (1,1)). Por lo tanto, se eliminan las reglas Sale_Hacia_Adelante, Sale_Hacia_CarrilDer_ConPrioridad y Sale_Hacia_CarrilDer_SinPrioridad. Además la regla Llega_Desde_CarrilDer:

Nuevo Estado	Estado del vecindario

1	(0,0) = 0 and (0,-1) = 0 and (-1,-1) = 1 and (-1,0) = 1

cambia por:

Nuevo Estado	Estado del vecindario
1	(0,0) = 0 and (0,-1) = 0 and (-1,-1) = 1

La modificación de esta regla consiste en no chequear por el estado del vecino (-1,0) pues éste se encuentra dentro de la obra, cuando un auto avanza desde (-1,-1). Luego, el resto de las reglas no cambian.

El conjunto de celdas con movimiento hacia adelante y en diagonal izquierda se define como:

$$\text{Celdas\_ADI} = \{ (\text{fil} + i, \text{col} + j) / -(\#n-1)/2 \le j \le -1 \wedge i = -\left(\frac{\#n-1}{2}\right) + |j| - 2 \ \}$$

La función τ para estas celdas se obtiene modificando el comportamiento normal definido para los vehículos en los tramos. Desde estas celdas los autos pueden avanzar derecho o utilizando la diagonal izquierda (hacia el vecino (1,1)). Por lo tanto, se eliminan las reglas Sale_Hacia_CarrilDer_ConPrioridad y Sale_Hacia_CarrilDer_SinPrioridad. Luego, el resto de las reglas no cambian.

El conjunto de celdas con movimiento en diagonal derecha se define como:

$$\text{Celdas\_DD} = \{ (\text{fil} + i, \text{col} + j) / -(\#n-1)/2 \le j \le -1 \wedge i = \left(\frac{\#n-1}{2}\right) - |j| + 1 \ \}$$

La función τ para estas celdas se obtiene modificando el comportamiento normal definido para los vehículos en los tramos. Desde estas celdas los autos sólo pueden avanzar usando la diagonal derecha (hacia el vecino (-1,1)). Por lo tanto, se eliminan las reglas Sale_Hacia_Adelante, Sale_Hacia_CarrilIz. Además las reglas Llega_Desde_CarrilIz_ConPrioridad y Llega_Desde_CarrilIz_SinPrioridad:

Nuevo Estado	Estado del vecindario
1	(0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1
1	(0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1 and ((2,-1) = 1 or (2,0) = 1)

cambia por:

Nuevo Estado	Estado del vecindario
1	(0,0) = 0 and (0,-1) = 0 and (1,-1) = 1

La modificación de esta regla consiste en no chequear por el estado del vecino (1,0) pues éste se encuentra dentro de la obra, cuando un auto avanza desde (1,-1). Tampoco se verifica la condición ((2,-1) = 1 or (2,0) = 1) pues ambos vecinos están dentro de la obra para todas estas celdas a excepción de una cuyo caso se analiza a continuación. Luego, el resto de las reglas para estas celdas no cambian.

Dentro de las celdas que sólo pueden avanzar en diagonal derecha, la celda (fil + 1,col - (n - 1) / 2) puede recibir vehículos que tienen los 2 movimientos en diagonal; mientras que las demás no. Por esto, para la celda (fil + 1,col - (n - 1) / 2) cambian las reglas Llega_Desde_CarrilIz_ConPrioridad y Llega_Desde_CarrilIz_SinPrioridad:

Nuevo Estado	Estado del vecindario
1	(0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1
1	(0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and (1,0) = 1 and ((2,-1) = 1 or (2,0) = 1)

por:

Nuevo Estado	Estado del vecindario
1	(0,0) = 0 and (0,-1) = 0 and (1,-1) = 1
1	(0,0) = 0 and (0,-1) = 0 and (1,-1) = 1 and ((2,-1) = 1 or (2,0) = 1)

respectivamente.
Esta modificación permite chequear que el vehículo no pueda avanzar en diagonal izquierda y por eso avanza hacia la celda origen. Las reglas establecen que los vehículos siempre intentan moverse hacia la izquierda antes de probar con la derecha.
Para las demás celdas del conjunto Celdas_DD no se chequea la condición ((2,-1) = 1 or (2,0) = 1), pues esas celdas forman parte de la obra y el vehículo no podrá avanzar hacia ellas (diagonal izquierda).

El conjunto de celdas con movimiento hacia adelante y en diagonal derecha se define como:

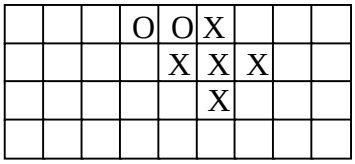
$$\text{Celdas\_ADD} = \{ \text{(fil + i, col + j)} / -(\#n-1)/2 \le j \le -1 \wedge i = \left(\frac{\#n-1}{2}\right) - |j| + 2 \ \}$$

La función τ para estas celdas se obtiene modificando el comportamiento normal definido para los vehículos en los tramos. Desde estas celdas los autos pueden avanzar derecho o utilizando la diagonal derecha (hacia el vecino (-1,1)). Por lo tanto, se elimina la regla Sale_Hacia_CarrilIz. Luego, el resto de las reglas no cambian.

En el segundo caso, para representar la obra o = (t, ni, ℓ, #n), se considera que el rombo toca el borde superior del tramo, es decir, ni = 0. Lo que cambia respecto al caso anterior es que al conjunto de las celdas de la obra se agregan las siguientes:

$$C_{TSup} = \{ \text{(fil - (\#n-1)/2 + i, col - (\#n-1)/2 + j)} / -(\#n-1)/2 \le j \le (\#n-1)/2 \wedge 0 \le i \le \left(\frac{\#n-1}{2}\right) - |j| \ \} = \{\text{(fil - (\#n-1)/2 + i, col - (\#n-1)/2 + j)} / |j| \le (\#n-1)/2 \wedge 0 \le i \le \left(\frac{\#n-1}{2}\right) - |j| \ \}$$

Esto se muestra en la siguiente figura:



X representa celda dentro de la obra
O representa celda agregada a la obra

Figura 37 – Obras sobre el borde superior del tramo

En este caso también se modifican las celdas que se encuentran delante de la obra, estas son:

$$\text{Celdas\_DD} = \{ (\text{fil} + i, \text{col} + j) / -\#n \le j \le -1 \wedge i = \left(\frac{\#n-1}{2}\right) - |j| + 1 \ \}$$

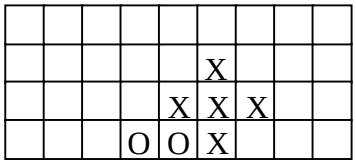
$$\text{Celdas\_ADD} = \{ (\text{fil} + i, \text{col} + j) / -\#n \le j \le -1 \wedge i = \left(\frac{\#n-1}{2}\right) - |j| + 2 \ \}$$

Para cada tipo de celda se establece el mismo comportamiento descripto anteriormente, de acuerdo a los movimientos que les permiten realizar a los vehículos.

El último caso, para representar la obra o = (t, ni, ℓ, #n), considera que el rombo toca el borde inferior del tramo, es decir, ni + #n = t_1 (t_1 es el parámetro del modelo M_t, definido para el tramo t, que acota la cantidad de filas del espacio). Lo que cambia respecto a cuando no toca ningún borde es que al conjunto de las celdas de la obra se agregan las siguientes:

C_{TInf} = {(fil + (#n-1)/2 + i, col - (#n-1)/2 + j) / $-(\#n-1)/2 \le j \le (\#n-1)/2 \wedge -\left(\frac{\#n-1}{2}\right) + |j| \le i \le 0$ } = {(fil + (#n-1)/2 + i, col - (#n-1)/2 + j) / $|j| \le (\#n-1)/2 \wedge -\left(\frac{\#n-1}{2}\right) + |j| \le i \le 0$ }

(el centro del triángulo es (fil + (#n-1)/2, col - (#n-1)/2).
Esto se muestra en la siguiente figura:



X representa celda dentro de la obra
O representa celda agregada a la obra

Figura 38 – Obras sobre borde Inferior

En este caso también se modifican las celdas que se encuentran delante de la obra, estas son:

$$\text{Celdas\_DI} = \{(\text{fil} + i, \text{col} + j) / -\#n \le j \le -1 \wedge i = -\left(\frac{\#n-1}{2}\right) + |j| - 1 \ \}$$

$$\text{Celdas\_ADI} = \{(\text{fil} + i, \text{col} + j) / -\#n \le j \le -1 \wedge i = -\left(\frac{\#n-1}{2}\right) + |j| - 2 \ \}$$

Para cada tipo de celda se establece el mismo comportamiento descripto anteriormente, de acuerdo a los movimientos que les permiten realizar a los vehículos.

Otra forma de modelar la presencia de obras, podría ser utilizando modelos no binarios y representando el estado de las celdas que forman parte de las obras con un valor distinto de 0 y 1. Así, cuando se tiene un vecino con este estado especial, nunca se intentará avanzar hacia él; esquivando las celdas con problemas. Con esto, se evita la necesidad de restringir el comportamiento de cada celda, como fuera planteado en esta sección. La reacción de las vecinas frente a una celda cuyo estado representa una obra, es semejante al de un vecino cercano a un choque. Este último se describe en la sección 2.4, donde se utilizan modelos no binarios para representar choques de vehículos.

1.4.4. Baches

Los baches se pueden especificar como:

$$Baches_T = \{ (t, n1, \ell) / t \in Tramos \wedge t = (c1, c2, n, a, dir, max) \wedge n1 \in [0, n-1] \wedge \ell \in N \}$$

Cada tupla del conjunto, b = (t, n1, ℓ), identifica el tramo (t) y el carril (n1) donde se encuentra el bache; y el desplazamiento del bache sobre el carril, es decir la distancia sobre el carril n1 que existe entre el bache y el comienzo del tramo (representada por ℓ). Por ejemplo, la Figura 39 muestra gráficamente la especificación de un bache.

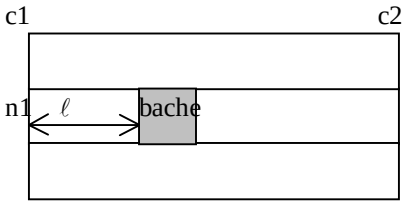


Figura 39 – Tramo con bache

Para que un bache (t, n1, ℓ) esté bien definido se debe cumplir que ℓ, sea menor o igual a la longitud del tramo. Esta restricción se escribe como:

$$\forall((c1, c2, n, a, dir, max), n1, l) \in Baches : 0 \leq l \leq Long\_Recta(c1, c2)$$

donde Long_Recta es una función definida en el Apéndice A y dado un par de puntos del plano calcula la distancia que hay entre ellos.

Para representar un bache en el modelo Cell-DEVS correspondiente al tramo, sólo se modifica el comportamiento de la celda que lo contiene. Para ello se utiliza una demora de transporte fija suficientemente grande que refleje que los vehículos circulan despacio debido a que la calle está rota. Un bache tendrá el tamaño de una celda y para modelar la presencia de uno más grande se deben definir varios consecutivos.

Dado un bache b = (t, n1, ℓ), se debe determinar a qué celda del tramo afecta. Esto se logra calculando la columna que representa la distancia ℓ y para ello basta dividir ℓ por la longitud de la celda, es decir

$$col = \lceil \ell / long\_celda \rceil$$

Por lo tanto la celda con bache es (n1, col) y su demora se define como:

$$d = d_{bache}$$

donde d_{bache} es una constante más grande que los valores de demora promedio de las celdas comunes. El resto de los parámetros que definen el comportamiento de la celda y el modelo del tramo t (M_t) no cambian.

Un bache también puede ser definido sobre un cruce, especificándolo como:

$$Baches_C = \{ c / c \in Cruces \}$$

Un bache en un cruce c, se modela eligiendo una posición al azar que tendrá demora alta y fija (d_{bache}). Para poder determinar su ubicación se necesita conocer la cantidad de celdas que tiene el cruce, que se obtiene del modelo definido para c:

$$M_c = < Xlist_c, Ylist_c, I_c, X_c, Y_c, n_c, t_{1c}, \eta_c, N_c, C_c, B_c, Z_c, select_c >$$

El parámetro buscado es t_{1c}. Entonces se elige un número entero al azar del intervalo [0, t_{1c} -1] que representará a la celda con bache cuya demora se define como:

$$d = d_{bache}.$$

De acuerdo a esta especificación de los baches a lo sumo se puede definir uno sobre cada cruce. Cuando hay muchos vehículos circulando, la celda con demora alta, provoca una reducción general de la velocidad en el anillo. Esto se debe a que los autos deben esperar a que se vacíe la celda de adelante, hasta llegar a la del bache que tarda bastante tiempo en hacerlo, provocando una espera en cadena.

Otra forma de representar que el cruce c tiene un bache, es hacer que todas las celdas de su modelo (M_c) tengan demora alta y fija (d_{bache}). Esto es definir la demora como $d = d_{bache}$, para todas las celdas; provocando que los vehículos disminuyan su velocidad al atravesar un cruce con bache.

1.4.5. Señales de tránsito y otros elementos de control

Las elevaciones transversales (lomo de burro), depresiones transversales (badén), bocacalles, irregularidades continuas (serrucho) y señales de PARE o de Escuela; se pueden especificar como:

ElementosDeControl = { (t, e, ℓ) / t $\in$ Tramos $\wedge$ $\ell \in$ N $\wedge$ e $\in$ {elevación transversal, depresión transversal, bocacalles, irregularidad continua, señal de PARE, señal de Escuela} }

Cada tupla del conjunto, ec = (t, e, ℓ), identifica el tramo (t = (c1, c2, n, a, dir, max)), el tipo de elemento de control se ha especificado (e) y la distancia entre el elemento de control y el comienzo del tramo (representada por ℓ). Por ejemplo, la Figura 40 muestra gráficamente la especificación de uno de estos elementos.

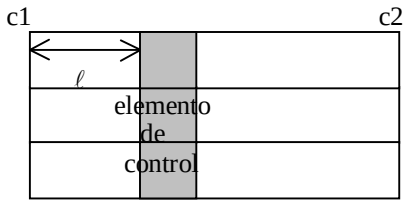


Figura 40 – Tramo con elemento de control

Para que un elemento de control (t, e, ℓ) esté bien definido se debe cumplir que ℓ, sea menor o igual a la longitud del tramo. Esta restricción se escribe como:

$$\forall((c1, c2, n, a, dir, max), e, l) \in ElementosDeControl : 0 \leq l \leq Long\_\text{Re}cta(c1, c2)$$

donde Long_Recta es una función definida en el Apéndice A y dado un par de puntos del plano calcula la distancia que hay entre ellos.

Como todos estos elementos de control se especifican y mapean sobre los modelos Cell-DEVS en forma análoga, se definen en forma conjunta estableciendo una individualización sólo cuando sea necesario. Una de las características comunes es que su presencia afecta a los vehículos de todos los carriles de la columna donde se encuentra. Además se definen sobre

tramos y no cruces, algunos se ubican comúnmente en la cercanías de las esquinas (bocacalles, señal de pare) pero ninguno tiene sentido dentro la intersección.

Para reflejar la presencia de cualquiera de estos elementos de control en el modelo del tramo t, se modifica el comportamiento de las celdas de todos los carriles donde se encuentra, utilizando una demora grande y fija. Cabe destacar que el objetivo de estos elementos de control es que los vehículos disminuyan la velocidad, y esto se logra aumentando la demora de las celdas. El tipo de demora (transporte o inercial) no cambia pero según el elemento de control se definirán distintas duraciones.

Dado un elemento de control ec = (t, e, ℓ), se debe determinar a qué columna del espacio de celdas del tramo afecta y para ello basta dividir ℓ por la longitud de la celda, es decir

$$col = \lceil \ell / long\_celda \rceil$$

Por lo tanto las celdas afectadas por el elemento de control ec son:

$$C_{ec} = \{ (i, col) / 0 \leq i \leq n-1 \}$$

donde n es la cantidad de carriles del tramo t. La demora para estas celdas se define como:

$$d = d_e$$

donde d_e es una constante que depende del elemento de control que se está modelando y es más grande que los valores de demora promedio de las celdas comunes. Cada elemento de control e puede tener una duración de demora diferente, representada por d_e. El resto de los parámetros que definen el comportamiento de la celda y el modelo del tramo t (M_t), no cambian.

La representación de los elementos de control y de los baches asignan una demora fija a las celdas que afectan y su superposición generaría problemas para determinar cuál es la nueva demora. De ahí, surge la restricción de que no se pueda definir más de un elemento de control (elevaciones o depresiones transversales, bocacalles, irregularidades continuas y señales de PARE o de Escuela) en la misma posición de un tramo, ni tampoco superponer alguno de ellos con la presencia de baches. En cambio pueden ser definidos cerca, siempre y cuando la distancia que los separa sea mayor que la longitud de celda (long_celda). Esta restricción se puede escribir como:

$$\forall (t_1, n_1, l_1) \in Baches_T, (t_2, e_2, l_2) \in ElementosDeControl :$$
$$\left(t_1 = t_2 \Rightarrow | l_1 - l_2 | > long\_celda\right)$$

$$\forall (t_1, e_1, l_1), (t_2, e_2, l_2) \in ElementosDeControl :$$
$$\left(\left(t_1 = t_2 \wedge | l_1 - l_2 | \leq long\_celda\right) \Rightarrow e_1 = e_2\right)$$

1.4.6. Autos estacionados

Los tramos con vehículos que estacionan sobre los carriles del borde, se pueden especificar como:

AutosEstacionados = { (t, n1) / t ∈ Tramos ∧ n1 ∈ {0, 1} ∧ t = (c1, c2, n, a, dir, max) ∧ n > 1 }

Cada tupla del conjunto, ce = (t, n1), identifica el tramo (t = (c1, c2, n, a, dir, max)) y el carril sobre el que estacionan los vehículos (representado por n1). Si n1 = 0, se estaciona sobre el carril 0 (carril izquierdo), si n1 = 1 lo hacen sobre el carril n-1 (carril derecho). Por lo tanto, el conjunto AutosEstacionados especifica calles con alguno o ambos bordes de estacionamiento e impone la restricción de que el tramo tenga por lo menos 2 carriles (n > 1), uno que es donde estacionan y el otro para la circulación de tráfico.

Para simplificar la definición del comportamiento de los vehículos no se permite estacionar en los 2 bordes de la calle si ésta no tiene más de 3 carriles, porque habría que definir un comportamiento diferente para algunas celdas del espacio desde las que los coches pueden estacionar hacia ambos lados. Esta restricción no se aleja demasiado de la realidad pues siempre deberían quedar varios carriles centrales de circulación de tráfico. Esto se puede escribir como:
$\forall t \in Tramos$:

$$\left(t = (c1, c2, n, a, dir, max) \wedge \{(t,1),(t,0)\} \subseteq AutosEstacionados\right) \Rightarrow (n > 3)$$

Cuando un vehículo avanza hacia uno de los carriles de estacionamiento, quedará detenido por un cierto tiempo. Esto se modela eligiendo la demora aleatoria para su velocidad, sobre un conjunto de valores grandes que representen varios minutos, horas o días. Así el vehículo permanece en la celda durante un período largo antes de continuar avanzando y cuando quiera arrancar para volver a circular, su movimiento de salida se hace en diagonal hacia adelante. Los autos que avanzan hacia carril de estacionamiento sólo provienen del contiguo, es decir, los autos sobre este el carril no pueden moverse derecho hacia adelante. Por lo tanto, en el modelo Cell-DEVS definido para el tramo se modifica la función que calcula la demora de las celdas de los carriles donde se modela el estacionamiento de vehículos y las reglas de movimiento de los vehículos de ellas y de las de los 2 carriles contiguos al que estacionan.

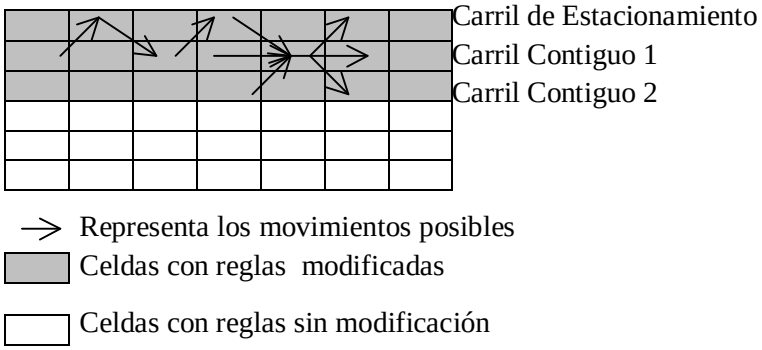


Figura 41 – Tramo con Carriles de Estacionamiento

Dado un carril definido para que estacionen los vehículos, ce = (t, n1), y sea M_t el modelo especificado para el tramo t:

$$M_t = < Xlist_t, Ylist_t, I_t, X_t, Y_t, n_t, \{t_1,...,t_n\}_t, \eta_t, N_t, C_t, B_t, Z_t, select_t >$$

Las celdas donde estacionan los autos y las de los carriles contiguos son:

$C_{est} = \{ (0, i) / 0 \le i \le t_2\text{-}1 \}$ si n1 = 0, ó $C_{est} = \{ (t_1\text{-}1, i) / 0 \le i \le t_2\text{-}1 \}$ si n1 = 1;

$C_{cont1} = \{ (1, i) / 0 \le i \le t_2\text{-}1 \}$ si n1 = 0, ó $C_{cont1} = \{ (t_1\text{-}2, i) / 0 \le i \le t_2\text{-}1 \}$ si n1 = 1; y

$C_{cont2} = \{ (2, i) / 0 \le i \le t_2\text{-}1 \}$ si n1 = 0, ó $C_{cont2} = \{ (t_1\text{-}3, i) / 0 \le i \le t_2\text{-}1 \}$ si n1 = 1;

respectivamente.

A continuación se define el comportamiento para las celdas de C_{est}, C_{cont1} y C_{cont2}, que es lo único que cambia del modelo del tramo (M_t) por el agregado de ce. Para estos conjuntos de celdas se restringen algunos de los movimientos que tenían los vehículos y para C_{est} también se modifica la demora.

Las celdas del carril de estacionamiento (C_{est}) no permiten que los vehículos avancen hacia adelante, sólo pueden llegar autos del carril contiguo y salir también hacia él. Para ellas se eliminan las reglas que modelan estos movimientos, que son Llega_Desde_Atrás y Sale_Hacia_Adelante. Luego el avance de un vehículo hacia estas celdas desde el carril contiguo ahora no debe chequear que se interponga en el camino de otro auto que avanza

derecho ((0,-1) = 0), porque ese movimiento fue eliminado. Por esto las reglas Llega_Desde_CarrilDer, Llega_Desde_CarrilIzConPrioridad y Llega_Desde_CarrilIzSinPrioridad:

Nuevo estado	Estado Vecindario	Nombre	Delay	d
1	(0,0) = 0 and (-1,-1) = 1 and (-1,0) = 1 and (0,-1) = 0	Llega_Desde_CarrilDer	transport	Conversión_Demora (velocidad (max))
1	(0,0) = 0 and (1,-1) = 1 and (1,0) = 1 and (0,-1) = 0	Llega_Desde_CarrilIz_ConPrioridad	transport	Conversión_Demora (velocidad (max))
1	(0,0) = 0 and (1,-1) = 1 and (1,0) = 1 and (0,-1) = 0 and ((2,-1) = 1 or (2,0) = 1)	Llega_Desde_CarrilIz_SinPrioridad	transport	Conversión_Demora (velocidad (max))

cambian por:

Nuevo estado	Estado Vecindario	Delay	d
1	(0,0) = 0 and (-1,-1) = 1 and (-1,0) = 1	transport	Conversión_Demora (estacionar)
1	(0,0) = 0 and (1,-1) = 1 and (1,0) = 1	transport	Conversión_Demora (estacionar)
1	(0,0) = 0 and (1,-1) = 1 and (1,0) = 1 and ((2,-1) = 1 or (2,0) = 1)	transport	Conversión_Demora (estacionar)

respectivamente.

Donde *estacionar* es una función aleatoria que devuelve un valor más grande que los que representan la velocidad normal de los vehículos, modelando que el auto ha estacionado.

El resto de las reglas para estas celdas no cambian. Entre ellas se encuentran las de ingreso y salida al cruce (Llega_Desde_Cruce y Sale_Hacia_Cruce) cuya demora no se modifica pues no se permite estacionar en las esquinas. Con los cambios descriptos sobre las reglas de estas celdas se modela que la celda origen sólo recibe autos de su vecino de atrás en diagonal (celda (-1,-1) ó (1,-1)) y que sólo se vacía si puede avanzar hacia adelante en diagonal (celda (-1,1) ó (1,1)); representando las maniobras de estacionamiento sobre el carril seleccionado.

Para las celdas del conjunto C_{cont1} sólo se modifican algunas reglas del movimiento de los vehículos. Dependiendo del parámetro n1, se introducen cambios en diferentes reglas. Si n1 = 0 (los autos estacionan sobre el carril 0), las celdas afectadas son:

$$C_{cont1} = \{ (1, i) / 0 \leq i \leq t_2\text{-}1 \}$$

Para estas celdas sólo es necesario redefinir las reglas Sale_Hacia_CarrilIz y Llega_Desde_CarrilIz_ConPrioridad que son las que envían/reciben autos hacia/desde el carril de estacionamiento. Estas reglas no deben chequear por la presencia de un auto estacionado que avance derecho porque ese movimiento ha sido eliminado. Así las reglas:

Nuevo estado	Estado Vecindario	Nombre	Delay	d
0	(0,0) = 1 and (1,1) = 0 and (1,0) = 0	Sale_Hacia_CarrilIz	transport	Conversión_Demora (velocidad (max))
1	(0,0) = 0 and (1,-1) = 1 and (1,0) = 1 and (0,-1) = 0	Llega_Desde_CarrilIz_ConPrioridad	transport	Conversión_Demora (velocidad (max))

cambian por:

Nuevo estado	Estado Vecindario	Delay	d
0	(0,0) = 1 and (1,1) = 0	transport	Conversión_Demora (velocidad (max))
1	(0,0) = 0 and (1,-1) = 1 and (0,-1) = 0	transport	Conversión_Demora (velocidad (max))

respectivamente.

Con estas modificaciones, cuando la celda origen recibe un auto del carril de estacionamiento no chequea si éste avanza derecho sobre su propio carril, pues siempre lo hará hacia ella (en diagonal). Además cuando la celda origen envia el auto hacia el carril de estacionamiento, no chequea que haya otro auto en ese carril, atrás del lugar para estacionar. El resto de las reglas para estas celdas no cambia.

Si n1 = 1 (los autos estacionan sobre el carril 1), las celdas del primer carril contiguo al de estacionamiento son:

$$C_{cont1} = \{ (t_1\text{-}2, i) / 0 \leq i \leq t_2\text{-}1 \}$$

Las modificaciones de las reglas para estas celdas se hacen en forma simétrica a las anteriores, es decir se redefinen las reglas Sale_Hacia_CarrilDer_ConPrioridad y Llega_Desde_CarrilDer, que son las que envían/reciben autos hacia/desde el carril de estacionamiento. Estas reglas no deben chequear por la presencia de un auto estacionado que avance derecho porque ese movimiento ha sido eliminado. Así las reglas:

Nuevo estado	Estado Vecindario	Nombre	Delay	d
0	(0,0) = 1 and (-1,1) = 0 and (-1,0) = 0	Sale_Hacia_CarrilDer_ConPrioridad	transport	Conversión_Demora (velocidad (max))
1	(0,0) = 0 and (-1,-1) = 1 and (-1,0) = 1 and (0,-1) = 0	Llega_Desde_CarrilDer	transport	Conversión_Demora (velocidad (max))

cambian por:

Nuevo estado	Estado Vecindario	Delay	d
0	(0,0) = 1 and (-1,1) = 0	transport	Conversió

			n_Demora (velocidad (max))
1	(0,0) = 0 and (-1,-1) = 1 and (0,-1) = 0	transport	Conversión_Demora (velocidad (max))

respectivamente.

El resto de las reglas para estas celdas no cambia.

Por último se describen los cambios de las reglas de las celdas del segundo carril contiguo al de estacionamiento. Dependiendo del parámetro n1, se introducen cambios en diferentes reglas. Si n1 = 0 (los autos estacionan sobre el carril 0), las celdas afectadas son:

$$C_{cont2} = \{ (2, i) / 0 \leq i \leq t_2\text{-}1 \}$$

Para estas celdas el único cambio que se introduce, es para la regla Llega_Desde_CarrilIz_SinPrioridad, en las celdas que estuviera definida. De esa regla se elimina la condición (2,-1) = 1, pues hace referencia al carril de estacionamiento y estaría chequeando que no haya un avance recto (movimiento que fuera eliminado). El resto de esta regla y todas las demás reglas para estas celdas no se modifican.

Si n1 = 1 (los autos estacionan sobre el carril 1), las celdas del segundo carril contiguo al de estacionamiento son:

$$C_{cont2} = \{ (t_1\text{-}3, i) / 0 \leq i \leq t_2\text{-}1 \}$$

Las modificaciones de las reglas para estas celdas se hacen en forma simétrica a las anteriores, eliminando la condición (-2,1) = 0 de la regla Sale_Hacia_CarrilDer_SinPrioridad. Ella hace referencia al carril de estacionamiento y estaría chequeando que no haya un avance recto (movimiento que fuera eliminado). El resto de esta regla y todas las demás reglas para estas celdas no se modifican.

Si el tramo tiene menos de 6 carriles y se definen vías de estacionamiento en los 2 bordes, algunas celdas del tramos pertenecerán a más de uno de los conjuntos definidos anteriormente (C_{est}, C_{cont1} y C_{cont2}). Teniendo en cuenta la restricción de sólo permitirlo cuando hay más de 3 carriles, no es posible que las celdas de C_{est} estén en otro de los conjuntos. Esto surge porque ellas modifican reglas que coinciden con los otros conjuntos. En cambio, puede suceder que la intersección de los conjuntos de celdas C_{cont1} y C_{cont2} de cada borde no sea vacía, en este caso se deben realizar las modificaciones correspondientes de las reglas señaladas para cada conjunto y no trae mayores inconvenientes pues se trata de reglas distintas las que deben cambiar.

1.4.7. Tasa de Choques

Una medición que se puede realizar es la tasa de choques producidos. Este valor se puede obtener revisando el log de salida de la simulación y contando las llegadas de distintos vehículos a una misma celda con diferencia de tiempo muy pequeña. La ventaja de hacer este cálculo a posteriori es reducir el procesamiento utilizado durante la ejecución de la simulación.

La detección de choques también se podría realizar en tiempo de ejecución, acoplando un modelo DEVS a cada celda que mida el tiempo transcurrido entre la llegada de los vehículos, y si éste es pequeño se ha producido un choque. Esto genera demasiado overhead comparado con la obtención de los datos del log. Pero si además de contar la cantidad de choques, se desea modelar el efecto de éste sobre las calles (secciones deshabilitadas), tal vez sea útil considerar

este enfoque. Aunque una solución menos costosa podría ser generar choques en forma aleatoria, de acuerdo a la tasa calculada en ejecuciones anteriores.

1.4.8. Marco Experimental

Los tramos que permiten el ingreso y salida de vehículos de la simulación se obtienen como:

$$\text{TramosIngreso} = \{ t \ / \ t = (p1, p2, n, a, dir, max) \wedge t \in \text{Tramos} \wedge [(dir = 0 \wedge (O\ v \in N : (p2,v) \in \text{Cruces})) \vee (dir = 1 \wedge (O\ v \in N : (p1,v) \in \text{Cruces}))] \}$$

$$\text{TramosSalida} = \{ t \ / \ t = (p1, p2, n, a, dir, max) \wedge t \in \text{Tramos} \wedge [(dir = 0 \wedge (O\ v \in N : (p1,v) \in \text{Cruces})) \vee (dir = 1 \wedge (O\ v \in N : (p2,v) \in \text{Cruces}))] \}$$

TramosIngreso y TramosSalida son subconjuntos de Tramos y se caracterizan por no tener un cruce de calles en alguno de sus extremos, el cual actuará como punto de ingreso o salida de vehículos de la simulación. Por lo tanto, estos conjuntos se derivan a partir de Tramos y Cruces.

Observación:
Si $t \in \text{Tramos} \wedge t = (p1, p2, n, a, dir, max)$ entonces se tienen los siguientes casos:

- los 2 extremos de t son cruces de calles: $(E\ v1,v2 \in N : (p1,v1),(p2,v2) \in \text{Cruces})$. En este caso $t \notin$ (TramosIngreso U TramosSalida).
- ninguno de los extremos de t es un cruce: $(O\ v1 \in N : (p1,v1) \in \text{Cruces}) \wedge (O\ v2 \in N : (p2,v2) \in \text{Cruces})$. En este caso $t \in \text{TramosIngreso} \wedge t \in \text{TramosSalida}$.
- sólo uno de los extremos de t es cruce: $((E\ v \in N : (p1,v) \in \text{Cruces}) \wedge dir = 0) \vee ((E\ v \in N : (p2,v) \in \text{Cruces}) \wedge dir = 1)$. En este caso $t \in \text{TramosIngreso} \wedge t \notin \text{TramosSalida}$.
- sólo uno de los extremos de t es cruce: $((E\ v \in N : (p2,v) \in \text{Cruces}) \wedge dir = 0) \vee ((E\ v \in N : (p1,v) \in \text{Cruces}) \wedge dir = 1)$. En este caso $t \in \text{TramosSalida} \wedge t \notin \text{TramosIngreso}$.

Cada tramo de los conjuntos TramosIngreso y TramosSalida se acopla con un modelo DEVS, encargado de generar o eliminar coches de la simulación, respectivamente. A continuación se comienza con la definición del modelo Generador y de Salida, luego se describe el modelo de los tramos de estos conjuntos y por último el acoplamiento.

Para cada tramo de ingreso $t_i \in$ TramosIngreso, se define un modelo DEVS encargado de generar vehículos que ingresan a la simulación. Aquí:

$$\text{Generador (\#c)} = < I, X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D >$$

donde #c indica la cantidad de carriles de t_i.

I = { $(\text{y-ti-auto}_j, \text{binario}) / 0 \leq j < \#c$ }
Cada port se acopla con un carril del tramo.

$X = \varnothing$

$Y = \{0, 1\}$
/* Representa la presencia (1) o ausencia de vehículos (0) */

S :
Variables Descriptivas
hay_auto (número binario)

carril (número natural)
La variable carril indica cuál es el próximo carril por el que ingresa el auto en la simulación.
Inicialización: hay_auto = 1; carril = 0; σ = tgen; phase = activa.

Parámetro
tgen (entero o real positivo), indica la frecuencia con que salen o se generan los vehículos que serán inyectados en la simulación.

δ_{ext}(s,e,x)
{
/* No hace nada */
}

λ(s)
{
send hay_auto *to* y-ti-auto$_{carril}$ /*envía señal que un auto quiere ingresar al tramo (1) o que no*/
/*hay nadie que quiera ingresar (0)*/
}

δ_{int}(s,e)
{
case phase
activa:
case hay_auto
0:
hay_auto = 1 /* planifica partida de un auto luego de tgen */
σ = tgen
phase = activa
carril = (carril + 1) mod #c
1:
hay_auto = 0 /* planifica evento para llevar a estado 0 */
σ = 0
phase = activa
end case
pasiva:
/* No hace nada */
end case
}

Este modelo genera un vehículo cada cierto tiempo (tgen) que ingresa a la simulación. Los autos generados entran por el siguiente carril respecto al último ingreso. De esta forma se van generando autos en cada carril del tramo en forma rotativa.

Para cada tramo de salida $t_s \in$ TramosSalida, se define un modelo DEVS encargado de eliminar vehículos de la simulación. Aquí:

Salida (#c) = < I, X, S, Y, δ_{int}, δ_{ext}, λ, D >

donde #c indica la cantidad de carriles de t_s.

I = { (x-ts-auto$_j$, binario) / $0 \leq j < \#c$ }
Cada port se acopla con un carril del tramo.

X = {0, 1}
/* Representa la llegada (1) o ausencia de vehículos (0) */

Y = ∅

S :
Variables Descriptivas
cant (número natural)
La variable cant permite llevar la cuenta de la cantidad de vehículos que abandonaron la simulación por esta salida.
Inicialización: cant = 0; phase = pasiva.

δ_{ext}(s,e,x)
{
cuando recibe x en el port x-ts-auto$_j$ $(0 \leq j < \#c)$
cant = cant +1
}

λ(s)
{
/* No hace nada */
}

δ_{int}(s,e)
{
/* No hace nada */
}

Este modelo recibe los vehículos que abandonan la simulación desde los tramos de salida y los cuenta.

Los tramos de ingreso y salida se modelan de la misma forma que los del conjunto Tramos con cruces en los extremos, pero cambia la interface del modelo pues no se acopla a cruces sino a un Generador o una Salida. Por lo tanto también cambian las reglas Sale_Hacia_Cruce y Llega_Desde_Cruce.

Para cada tramo de ingreso $t_i \in$ TramosIngreso, se eliminan los ports externos

{ <$Y_{\eta+1}$(j,0), binario> / $0 \leq j < n$ }

donde n es la cantidad de carriles de t_i.

La regla Llega_Desde_Cruce, no cambia pues recibe un 1 cuando se genera un nuevo auto.

Nuevo estado	Estado Vecindario	Nombre	Delay
1	(0,0) = 0 and portvalue($X_{\eta+1}$) = 1	Llega_Desde_Cruce	transport

Esta regla ahora representa que el modelo Generador asociado al tramo ha producido un nuevo auto. El Generador no chequea el estado de la celda del tramo, pues si el tramo no tiene lugar para el nuevo auto, lo ignora.

Para cada tramo de salida $t_s \in$ TramosSalida, se eliminan los ports externos:

$$\{ <X_{\eta+1}(j,k-1), \text{binario}> / \ 0 \leq j < n \}$$

donde n es la cantidad de carriles de t y k la cantidad de celdas.

La regla de salida para el cruce era:

Nuevo estado	Estado Vecindario	Nombre	Delay
0	(0,0) = 1 and portvalue(x-c-haylugar) = 0	Sale_Hacia_Cruce	inercial

y cambia por:

Nuevo estado	Estado Vecindario	Delay
0	(0,0) = 1	transport

Esta nueva regla no chequea por que haya lugar para poder salir del tramo. Una vez que un auto llega a ella, saldrá de la simulación luego de la demora correspondiente.

Para establecer los acoplamientos entre tramos de ingreso/salida y los modelos generadores/salidas se debe considerar que, para cada elemento ti del conjunto TramosIngreso se construye un modelo Generador (M_G). Además, para cada elemento ts del conjunto TramosSalida se construye un modelo Salida (M_S). Así, cada tramo ti, ts genera las siguientes influencias:

/* Definición de las influencias $\{I_i\}$*/
$I_G = \{ M_{ti} \}$
$I_{ts} = \{ M_S \}$

El acoplamiento de los modelos se define como:

Para cada $t_i \in$ TramosIngreso con t = (c1, c2, n, a, dir, max) y k = Ctd_Celdas(t)

$$Z_{Gti} : (\text{y-ti-auto}_j)_G \rightarrow X_{\eta+1}(j, 0)_{ti}, \forall (j \in N, j \in [0, n-1])$$

Para cada $t_s \in$ TramosSalida con t = (c1, c2, n, a, dir, max) y k = Ctd_Celdas(t)

$$Z_{tsS} : Y_{\eta+1}(j, k-1)_{ts} \rightarrow (\text{x-ts-auto}_j)_S, \forall (j \in N, j \in [0, n-1])$$

2. Extensión del Lenguaje

Aquí se extiende el lenguaje de especificación de secciones de ciudad mediante la incorporación de calles con circulación de camiones y choques de vehículos. Esto será modelado con espacios Cell-DEVS cuyo estado se representa con un número natural, a diferencia de los modelos binarios planteados en la sección 1.

2.3. Camiones

Para representar los camiones en el modelo de tráfico se definen nuevas calles y cruces que permitan diferenciarlos de los autos; siguiendo los mismos lineamientos planteados para la circulación exclusiva de coches.

2.3.1. Calles con circulación de camiones

Los tramos con circulación de camiones se pueden especificar como:

$TramosCamiones = \{ (p1, p2, n, a, dir, max) / p1, p2 \in Puntos \wedge n, max \in N \wedge a, dir \in \{0, 1\} \}$

Cada elemento de este conjunto es una tupla de 6 componentes, cuyo significado es el mismo que fuera definido para los tramos de la sección 1.3.1 (exclusivos para autos). La diferencia entre ambos se verá reflejada en sus modelos Cell-DEVS.

Para cada tramo $t = (p1, p2, n, a, dir, max)$, $t \in TramosCamiones$, se define un espacio Cell-DEVS con demora de transporte. A continuación se describe el modelo de las celdas de este espacio y luego se introduce el modelo acoplado. Cabe destacar que el comportamiento y vecindad depende del carril donde se encuentre la celda y de las líneas de tráfico totales del tramo. Aquí sólo se presentan estos conceptos en forma genérica; mientras que en el Apéndice E son detallados de acuerdo a la ubicación de las distintas celdas.

Cada celda del espacio del tramo t se define como:

$$C_{ij} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

$\boldsymbol{X} = \boldsymbol{Y} = N$;

S:

$$s = \begin{cases} 1 & \text{si hay un auto en la celda;} \\ k : k \equiv r \bmod 10 \wedge 2 \leq r \leq 5 & \text{si hay un camión en la celda;} \\ 0 & \text{si la celda está vacía.} \end{cases}$$

donde,
$k, r \in N$ y r es el resto de dividir k por 10;

delay = transport;

$\boldsymbol{\lambda}$, $\boldsymbol{\delta_{int}}$ y $\boldsymbol{\delta_{ext}}$ se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

Un cambio importante respecto a los tramos sin camiones es que el estado de las celdas que antes era binario ahora se representa con un número natural. El valor 0 (s = 0) sigue indicando que la celda vacía, el valor 1 (s = 1) la presencia de un auto en la celda y para los camiones se utilizan los estados del siguiente conjunto:

$$\{ s / s \in N \wedge s > 1 \wedge s \equiv r \bmod 10 \wedge 2 \leq r \leq 5 \}$$

Donde cada elemento representa un identificador único de un camión con longitud r. Ésta permite establecer cuál es el lugar libre necesario para cambiar de carril (tantas celdas como la longitud del camión deben estar vacías). El identificador único del camión se utiliza cuando 2 camiones se acercan, para poder distinguirlos entre sí y evitar ambigüedades en las reglas (como por ejemplo, que se confundan con un camión nuevo más largo). Además se impone la restricción de que los camiones tengan longitud entre 2 y 5 celdas, que es un valor razonable considerando que cada celda representa aproximadamente el tamaño de un auto. Esta cota se fija de antemano pues la longitud máxima de un camión determina la cantidad de vecinas hacia atrás que deben tener las celdas para saber si cuentan con suficiente espacio para el movimiento del camión completo.

El vecindario (N) se define como:

N = {(3,0), (2,-5), (2,-4), (2,-3), (2,-2), (2,-1), (2,0), (2,1), (1,-5), (1,-4), (1,-3), (1,-2), (1,-1), (1,0), (1,1), (1,2), (0,-5), (0,-4), (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2), (-1,-5), (-1,-4), (-1,-3), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2), (-2,-5), (-2,-4), (-2,-3), (-2,-2), (-2,-1), (-2,0), (-2,1) }

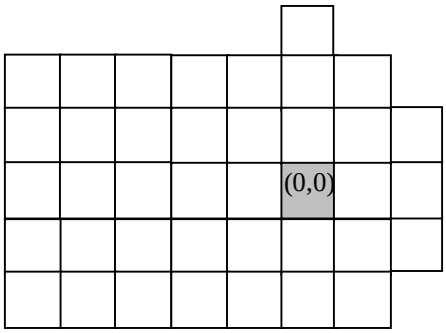


Figura 42 – Vecindario

Luego la interface de las celdas (I) se obtiene a partir de N, como fuera descripto por el formalismo Cell-DEVS.

Las reglas de comportamiento que definen a la función τ se dividen en 2 grupos, el primero contiene todas aquellas que representan el movimiento de los camiones y el segundo el desplazamiento de los autos. El movimiento de un camión implica el cambio de estado de varias celdas que reaccionan en forma conjunta. En particular, la celda de adelante de todo representa la trompa o cabeza del camión y es quien elige el próximo movimiento (hacia adelante, hacia izquierda o hacia derecha). Para poder tomar esa decisión debe tener un vecindario suficientemente grande que permita saber si todo el camión se podrá mover o no. El movimiento del resto de las celdas que forman parte del camión, sólo consiste en seguir a la parte del camión que tienen adelante; que la pueden identificar en forma unívoca a partir del estado de los vecinos. Además la demora o velocidad del camión también es responsabilidad de la celda que actua como cabeza, el resto se mueve con demora 0 para representar el movimiento conjunto del camión.

Se definen 3 movimientos para los camiones: avance recto, desplazamiento hacia derecha y desplazamiento hacia izquierda, que se grafican en la Figura 43. El primero consiste en avanzar desde una celda a la siguiente sobre el mismo carril. El segundo y tercero consisten en cambiar

hacia el carril de la derecha e izquierda, respectivamente; sin avanzar hacia adelante. Un camión siempre intenta moverse hacia adelante, en caso de no haber lugar lo hace hacia la derecha y por último lo intenta hacia la izquierda. Esto se modela así para representar que los camiones tienden a circular por los carriles de la derecha.

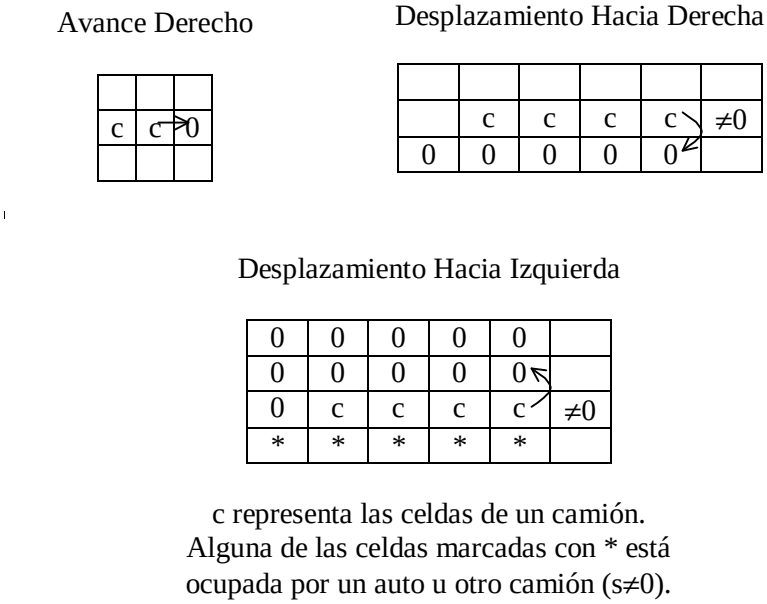


Figura 43 – Movimientos de los camiones

En los tramos circulan tanto camiones como autos, y es por ello que se debe definir alguna política que regule el movimiento de ambos. Cualquier vehículo que avance derecho sobre su propio carril tiene prioridad frente a los que quieran avanzar desde otros carriles, esto significa que para avanzar derecho sólo es necesario chequear que la posición de adelante esté vacía, sin importar si se trata de un camión o de un auto. Luego cuando un camión quiere moverse hacia la derecha, sólo chequea que haya lugar, pues este movimiento tiene prioridad frente al avance de un auto hacia esas mismas posiciones. Pero cuando un camión intenta moverse hacia la izquierda, debe ver que haya lugar y además que no haya otro vehículo (auto, camión) que quiera acceder a las mismas posiciones, pues este movimiento no tiene prioridad respecto al avance de autos o de camiones desde izquierda. Por lo tanto, para establecer el nuevo estado de la celda se requiere de un vecindario que permita chequear todos estos casos y se muestra en la Figura 42.

En definitiva, se obtienen las siguientes reglas que modelan el movimiento de los camiones:

Nuevo estado	Estado Vecindario	Delay	d	Nombre
(0,-1)	(0,0) = 0 and HayCamión(0,-1) and EsCabeza(0,-1)	transport	Conversión_Demora (veloc_camión(max))	Llega_Camión_Desde_Atrás
(0,-1)	(0,0) = 0 and HayCamión(0,-1) and (0,1) = (0,-1)	transport	0	

Donde,

- veloc_camión: N $\rightarrow$ N, es una función aleatoria que recibe como parámetro el valor máximo que puede alcanzar la función y retorna un valor natural que representa una velocidad en km/h elegida en forma aleatoria de acuerdo a la distribución característica.

- max es la constante especificada en el tramo y representa la velocidad máxima de circulación permitida, en km/h.
- Conversión_Demora es una función detallada en el Apéndice A, recibe como parámetro un valor natural que representa una velocidad (en km/h) y devuelve el tiempo (en segundos) que se debe demorar el auto en la celda para representar que avanza con esa velocidad.
- HayCamión(i,j) es una macro que verifica que en la posición haya una parte de algún camión. Es decir,

  HayCamión(i,j) ≡ remainder((i,j),10) <= 5 and remainder((i,j),10) >= 2
- EsCabeza(i,j) es una macro que se fija si la posición (i,j) es la cabeza o trompa del camión; para eso se chequea que en la posiciones de adelante no haya un vecino que tenga otra parte del mismo camión. Esto es:
  EsCabeza(i,j) ≡ (i,j+1) != (i,j) and (i,j+2) != (i,j) and (i+1,j+1) != (i,j) and (i-1,j+1) != (i,j)

	≠ (i,j)	
(i,j)	≠ (i,j)	≠ (i,j)
	≠ (i,j)	

Figura 44 – Cabeza del camión

La regla Llega_Camión_Desde_Atrás representa la llegada de un camión a la celda origen desde la celda de atrás. Se diferencia el caso en que lo que llega es la cabeza que es la única parte del camión con demora. Además el movimiento del resto del camión es seguir a la parte que tienen adelante, sin demora y sin decidir la dirección del próximo movimiento en función del espacio libre.

Nuevo estado	Estado Vecindario	Delay	d	Nombre
0	HayCamión(0,0) and (0,1) = 0 and EsCabeza(0,0)	transport	Conversión_Demora (veloc_camión(max))	Sale_Camión_Hacia_Adelante
0	HayCamión(0,0) and (0,1) = 0 and (0,2) = (0,0)	transport	0	

Esta regla representa la salida de un camión desde la celda origen hacia adelante, diferenciando como antes el caso de que se trate de la cabeza u otra parte del camión.

Nuevo estado	Estado Vecindario	Delay	d	Nombre
0	HayCamión(0,0) and EsCabeza(0,0) and CarrilDerLibre(0,0)	transport	Conversión_Demora (veloc_camión(max))	Sale_Camión_Hacia_CarrilDer
0	HayCamión(0,0) and (-1,1) = (0,0)	transport	0	

Donde,

- CarrilDerLibre(i,j) es una macro que verifica que en el carril derecho de la celda (i,j) haya suficiente lugar para que el camión se pueda desplazar hacia allí. Es decir,

  CarrilDerLibre(i,j) ≡ ∀k: 0 ≤ k ≤ remainder((i,j),10) ⇒ (i-1,j-k) = 0

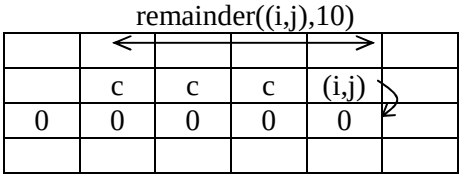


Figura 45 – Carril Derecho Libre

La regla Sale_Camión_Hacia_CarrilDer representa la salida de un camión desde la celda hacia la derecha (vecino (-1,0)), diferenciando como antes el caso de que se trate de la cabeza u otra parte del camión. Sólo la cabeza del camión chequea que haya suficiente lugar para el movimiento hacia la derecha de todo el camión (CarrilDerLibre(0,0)), el resto sólo sigue al de adelante.

Nuevo estado	Estado Vecindario	Delay	d	Nombre
(1,0)	(0,0) = 0 and HayCamión(1,0) and (1,1) != 0 and EsCabeza(1,0) and CarrilDerLibre(1,0)	transport	Conversión_Demora (veloc_camión(max))	Llega_Camión_Desde_CarrilIz
(1,0)	(0,0) = 0 and HayCamión(1,0) and (0,1) = (1,0)	transport	0	

Esta regla representa la llegada de un camión a la celda origen desde la izquierda. Si se trata de la cabeza del camión entonces se debe chequear que el movimiento elegido haya sido desplazarse hacia derecha, por esto se verifica que el camión no pueda avanzar hacia adelante ((1,1) != 0) y que tenga suficiente espacio para este movimiento (CarrilDerLibre(1,0)).

Nuevo estado	Estado Vecindario	Delay	d	Nombre
0	HayCamión(0,0) and EsCabeza(0,0) and 2CarrilesIzLibres(0,0)	transport	Conversión_Demora (veloc_camión(max))	Sale_Camión_Hacia_CarrilIz
0	HayCamión(0,0) and (1,1) = (0,0)	transport	0	

Donde,

- 2CarrilesIzLibres(i,j) es una macro que verifica que en los 2 carriles hacia la izquierda de la celda (i,j) haya suficiente lugar para que el camión se pueda desplazar hacia allí. Es decir, 2CarrilesIzLibres(i,j) ≡ (i,j-remainder((i,j),10)) = 0 and
  [∀k: 0 ≤ k ≤ remainder((i,j),10) ⇒ ((i+1,j-k) = 0 and (i+2,j-k) = 0)]

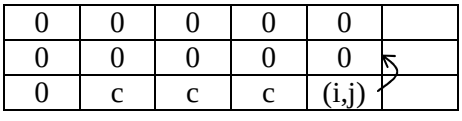


Figura 46 - Carriles Izquierdos Libres

La regla Sale_Camión_Hacia_CarrilIz representa la salida de un camión desde la celda hacia la izquierda (vecino (1,0)). Sólo la cabeza del camión chequea que haya suficiente lugar para el movimiento hacia la izquierda de todo el camión (2CarrilesIzLibres(0,0)), el resto sólo sigue al de adelante. Se verifica que hayan 2 carriles hacia izquierda libres porque cualquier otro camión o auto que pueda acceder a las mismas posiciones tiene prioridad de hacerlo.

Nuevo estado	Estado Vecindario	Delay	d	Nombre
(-1,0)	(0,0) = 0 and HayCamión(-1,0) and (-1,1) != 0 and EsCabeza(-1,0) and not(CarrilDerLibre(-1,0)) and 2CarrilesIzLibres(-1,0)	transport	Conversión_Demora (veloc_camión(max))	Llega_Camión_Desde_CarrilDer
(-1,0)	(0,0) = 0 and HayCamión(-1,0) and (0,1) = (-1,0)	transport	0	

Esta regla representa la llegada de un camión a la celda origen desde la derecha. Si se trata de la cabeza del camión entonces se debe chequear que el movimiento elegido haya sido desplazarse hacia izquierda, por esto se verifica que el camión no pueda avanzar ni hacia adelante ((1,1) != 0) ni hacia derecha (not(CarrilDerLibre(-1,0))); y que tenga suficiente espacio para este movimiento (2CarrilesIzLibres(-1,0)).

Luego, como en los tramos también circulan autos, se definen la siguientes reglas que modelan su movimiento:

Nuevo estado	Estado Vecindario	Delay	d	Nombre
1	(0,0) = 0 and (0,-1) = 1	transport	Conversión_Demora(velocidad(max))	Llega_Auto_Desde_Atrás

Donde,

- velocidad: N $\rightarrow$ N, es una función aleatoria con la distribución probabilística característica del flujo de vehículos. Se espera que pocos autos circulen con las velocidades extremas (máxima y mínima) y que la mayoría lo haga con velocidades intermedias. Recibe como parámetro el valor máximo que puede alcanzar la función y en base a él se obtiene la media $\bar{x} = \frac{2}{3} * max(km/h)$ y desvío estándar $\sigma = \frac{1}{3} * max(km/h)$, en este caso max es el parámetro pasado a la función. El valor que retorna es un natural que representa una velocidad en km/h elegida en forma aleatoria de acuerdo a la distribución descripta.
- max es la constante especificada en el tramo y representa la velocidad máxima de circulación permitida, en km/h.
- Conversión_Demora es una función detallada en el Apéndice A, recibe como parámetro un valor natural que representa una velocidad (en km/h) y devuelve el tiempo (en segundos) que se debe demorar el auto en la celda para representar que avanza con esa velocidad.

La regla Llega_Auto_Desde_Atrás representa la llegada de un auto a la celda origen desde la celda de atrás.

Nuevo estado	Estado Vecindario	Delay	d	Nombre

0	(0,0) = 1 and (0,1) = 0	transport	Conversión_Demora(velocidad(max))	Sale_Auto_Hacia_Adelante

Esta regla representa la salida de un auto desde la celda origen hacia adelante.

Nuevo estado	Estado Vecindario	Delay	d	Nombre
0	(0,0) = 1 and (1,0) = 0 and (1,1) = 0 and not(HayCamión(2,1))	transport	Conversión_Demora (velocidad (max))	Sale_Auto_Hacia_CarrilIz

Esta regla representa la salida de un auto desde la celda hacia la izquierda (vecino (1,1)). Se chequea que haya suficiente lugar para el movimiento ((1,0) = 0 and (1,1) = 0) y que no haya un camión que pueda acceder a la misma posición con prioridad de movimiento (not(HayCamión(2,1))).

Nuevo estado	Estado Vecindario	Delay	d	Nombre
1	(0,0) = 0 and (-1,-1) = 1 and (-1,0) != 0 and (0,-1) = 0 and not(HayCamión(1,0))	transport	Conversión_Demora (velocidad (max))	Llega_Auto_Desde_CarrilDer

Esta regla representa la llegada de un auto a la celda origen desde la derecha. Se debe chequear que el movimiento elegido haya sido desplazarse hacia izquierda, por esto se verifica que el auto no pueda avanzar hacia adelante ((-1,0) != 0), que tenga suficiente espacio para este movimiento ((0,-1) = 0 and (0,0) = 0) y que no haya un camión que pueda acceder a la misma posición con prioridad de movimiento (not(HayCamión(1,0))).

Nuevo estado	Estado Vecindario	Delay	d	Nombre
0	(0,0) = 1 and (-1,0) = 0 and (-1,1) = 0 and not(HayCamión(0,1)) and ((-2,0) = 0 or (-2,1) = 0)	transport	Conversión_Demora (velocidad (max))	Sale_Auto_Hacia_CarrilDer

Esta regla representa la salida de un auto desde la celda hacia la derecha (vecino (-1,1)). Se chequea que haya suficiente lugar para el movimiento hacia la derecha ((-1,0) = 0 and (-1,1) = 0), que no haya un camión que pueda acceder a la misma posición con prioridad de movimiento (not(HayCamión(0,1))) y que tampoco haya un auto con prioridad que quiera avanzar a la misma posición ((-2,0) = 0 or (-2,1) = 0).

Nuevo estado	Estado Vecindario	Delay	d	Nombre
1	(0,0) = 0 and (1,-1) = 1 and (1,0) != 0 and not(HayCamión(1,0)) and (0,-1) = 0 and ((2,-1) != 0 or (2,0) != 0 or HayCamión(3,0))	transport	Conversión_Demora (velocidad (max))	Llega_Auto_Desde_CarrilIz

Esta regla representa la llegada de un camión a la celda origen desde la izquierda. Se chequea que haya lugar sobre el carril para el avance del auto desde la izquierda ((0,0) = 0 and (0,-1) = 0), que el auto no se pueda mover derecho sobre su propio carril ((1,0) != 0) y que no pueda avanzar hacia su carril izquierdo ((2,-1) != 0 or (2,0) != 0 or HayCamión(3,0)). Además se chequea que no haya un camión con prioridad (not(HayCamión(1,0))) de movimiento hacia la celda origen.

En el Apéndice E se detalla la utilización de estas reglas (para el movimiento de autos y camiones) de acuerdo al vecindario de cada celda; diferenciando en casos según la cantidad de carriles del tramo.

El modelo acoplado correspondiente al tramo t = (p1, p2, n, a, dir, max) se define como:

$$TC(k, max, \#c) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z, select >$$

donde,

Ylist = { (i,0) / 0 ≤ i < #c } U { (i,k-1) / 0 ≤ i < #c }

Xlist = { (i,0) / 0 ≤ i < #c } U { (i,k-1) / 0 ≤ i < #c }

I = $<P^x, P^y>$

$P^x = \{<X_{\eta+1}(i,0), natural> / \ 0 \le i < \#c\} \cup \{<X_{\eta+1}(i,k-1), natural> / \ 0 \le i < \#c\}$
$P^y = \{<Y_{\eta+1}(i,0), natural> / 0 \le i < \#c\} \cup \{<Y_{\eta+1}(i,k-1), natural> / \ 0 \le i < \#c\}$

Estos ports serán denotados de la siguiente forma:

Port	Nombre	Comentario
$X_{\eta+1}(i,0)$, 0 ≤ i ≤ #c-1	x-c-hayvehículo	Este port se utiliza para informar de la salida de un vehículo desde el cruce hacia el tramo.
$X_{\eta+1}(i,k-1)$, 0 ≤ i ≤ #c-1	x-c-haylugar	Este port permite saber si en el cruce hay lugar (portvalue($X_{\eta+1}$) = 0) para que avance un vehículo desde el tramo.
$Y_{\eta+1}(i,0)$, 0 ≤ i ≤ #c-1	y-c-haylugar	Este port se utiliza para que el cruce pueda saber si hay lugar en el tramo para que un vehículo pueda salir de él (portvalue($Y_{\eta+1}$) = 0).
$Y_{\eta+1}(i,k-1)$, 0 ≤ i ≤ #c-1	y-c-hayvehículo	Este port informa al cruce de la presencia de un vehículo en el tramo que desea avanzar hacia él.

X = ***Y*** = N

n = 2

$\mathbf{t_1}$ = #c

$\mathbf{t_2}$ = k

C = { C_{ij} / i ∈ [0, #c-1] ∧ j ∈ [0, k-1] }, donde cada C_{ij} es un componente Cell-DEVS con demora y la especificación de cada celda ha sido descripta antes de introducir el modelo acoplado.

B = { (i,j) / 0 ≤ i ≤ #c-1 ∧ 0 ≤ j ≤ 5 } U { (i,j) / 0 ≤ i ≤ #c-1 ∧ k-2 ≤ j ≤ k-1 }

Z se construye siguiendo la definición dada por el formalismo Cell_DEVS, instanciada con el vecindario.

select se inicializa de igual forma que para los tramos sin camiones, de acuerdo a la cantidad de carriles.

TC(k, max, #c) significa Tramo de #c Carriles de longitud k (celdas) cada uno con velocidad máxima max (en Km/h). Donde max y #c es la velocidad máxima y cantidad de carriles especificada para t, respectivamente; y k = Ctd_Celdas(t). La función Ctd_Celdas se encuentra definida en el Apéndice A y establece el parámetro calculando la longitud del tramo (a partir de los extremos especificados para t), y luego dividiéndola por el tamaño definido para la celda.

El comportamiento de las celdas borde es diferente del definido para el resto del espacio. Las celdas de la columna 0 ({ (i,0) / 0 ≤ i ≤ #c-1 }) de los tramos serán las encargadas de recibir los camiones desde el cruce y expandirlos a su longitud (indicada por el estado), pues en él ocupan una sola celda. Para modelarlo se utilizan las siguientes celdas:

Nuevo estado	Estado Vecindario	Delay	d	Nombre
1	(0,0) = 0 and portvalue(x-c-hayvehículo) = 1	transport	Conversión_Demora (velocidad(max))	Llega_Auto_Desde_Cruce

Esta regla representa la llegada de un auto a la celda origen desde el cruce.

Nuevo estado	Estado Vecindario	Delay	d	Nombre
portvalue(x-c-hayvehículo)	(0,0) = 0 and EsCamión(portvalue(x-c-hayvehículo))	transport	Conversión_Demora (veloc_camión(max))	Llega_Camión_Desde_Cruce

Esta regla representa la llegada de un camión a la celda origen desde el cruce.

Las 4 reglas que se especifican a continuación serán denotadas como Reconstrucción_Camión:

Nuevo estado	Estado Vecindario	Delay	d	Nuevo Estado de los Ports
0	HayCamión(0,0) and (0,1) = 0 and CamiónCompleto(0,0)	transport	0	send(0,y-c-haylugar)
0	HayCamión(0,0) and (0,1) = 0 and not(CamiónCompleto(0,0)) and not(EsCabeza_TC1(0,0))	transport	0	send((0,0),y-c-haylugar)
(0,1)	(0,0) = 0 and HayCamión(0,1) and not(CamiónCompleto(0,1))	transport	0	send((0,1),y-c-haylugar)
0	HayCamión(0,0) and (0,1) = 0 and EsCabeza_TC1(0,0)	transport	Conversión_Demora (veloc_camión(max))	send((0,0),y-c-haylugar)

Donde,

- EsCamión(k) es una macro que verifica que el valor pasado como parámetro represente a un camión. Es decir,

EsCamión(k) ≡ remainder(k,10) <= 5 and remainder(k,10) >= 2

- CamiónCompleto(i,j) es una macro que se fija si el camión ha sido generado completo, es decir si la cantidad de vecinas con estado (i,j) es igual a la longitud del camión. Esto es:

  CamiónCompleto(i,j) ≡ (statecount((i,j)) = remainder((i,j),10))

- EsCabeza_TC1(i,j) es una macro que se fija si la posición (i,j) es la cabeza o trompa del camión; para eso se chequea que en las posiciones de adelante no haya un vecino que tenga otra parte del mismo camión. Esta macro tiene en cuenta que la celda (i,j) se encuentra en un tramo de único carril. Es decir,

  EsCabeza_TC1(i,j) ≡ (i,j+1) != (i,j) and (i,j+2) != (i,j)

Estas reglas representan la construcción del camión cuando sale del cruce pues allí ocupa una sola celda. La primera regla representa que el camión ya ha sido generado completamente, por lo que se envía al cruce que la celda está vacía y puede recibir otro vehículo. La segunda regla representa que una parte del camión avanza hacia la celda siguiente a la origen pero todavía falta generar otras partes del camión por lo que el cruce no debe percibir el vaciamiento momentáneo de la celda. Esto es para impedir que llegue desde el cruce otro vehículo en medio del camión que se está construyendo. La tercera regla representa la generación de una nueva parte del camión en la celda origen, siempre y cuando todavía no se haya reconstruido el camión completo. La última regla diferencia el caso en que en la celda origen se encuentra la cabeza del camión pues debe tener la demora correspondiente a la velocidad del mismo. Esta regla equivale a la segunda pero para el caso que se trate de la cabeza del camión.

Por ende, las celdas de la columna 0 cuentan con las reglas: Llega_Auto_Desde_Cruce, Llega_Camión_Desde_Cruce, Reconstrucción_Camión y las reglas Sale_Auto_Hacia_* que le corresponden según el carril donde se ubican. Además su vecindario se extiende hacia adelante incorporando las celdas { (0,k) / 3 ≤ k ≤ 5 }, que son necesarias para poder determinar cuándo se ha reconstruído el camión completamente.

Las celdas de las columnas 1, 2, 3, y 4 tienen comportamiento diferente pues no permiten que los camiones se muevan hacia los costados; el resto de las reglas son las mismas que le corresponden según el carril donde se encuentran. Es decir, mientras se expande el camión sólo se permite el movimiento derecho hacia adelante. Luego las celdas de la columna 5 sólo reciben camiones con movimiento derecho pero permiten la salida hacia cualquier carril adyacente (seguro que cuando llegan a estas celdas el camión ya está completo pues la longitud máxima posible es 5). Para estas celdas los movimientos de los autos son los mismos que le corresponden según el carril donde se encuentran.

Por otro lado, también son celdas borde aquellas ubicadas en el otro extremo del tramo. Las celdas de la columna k-1 son las que se comunican con los cruces y realizarán la compactación del camión. Como dentro del cruce cualquier camión ocupa una única posición, entonces las celdas de la columna k-1 sólo envían la presencia del camión por los ports externos cuando reciben su cabeza.

Las 3 reglas que se especifican a continuación serán denotadas como Compactación_Camión:

Nuevo estado	Estado Vecindario	Delay	d	Nuevo Estado de los Ports
0	HayCamión(0,0) and portvalue(x-c-haylugar) = 0	inercial	Conversión_Demora (veloc_camión(max))	send((0,0),y-c-hayvehículo)

(0,-1)	(0,0) = 0 and HayCamión(0,-1) and CamiónCompleto(0,-1)	transport	Conversión_Demora (veloc_camión(max))	send(0,y-c-hayvehículo)
0	(0,0) = 0 and HayCamión(0,-1) and not(CamiónCompleto(0,-1))	transport	0	send(0,y-c-hayvehículo)

Estas reglas representan la compactación del camión para que ingrese al cruce ocupando una sola celda. La primera de ellas es la que representa que el camión ingresa al cruce cuando tiene suficiente lugar. La celda origen sólo cambia de estado cuando contiene la cabeza del camión, para el resto se mantiene en estado 0. Por esto cada vez que hay un camión en la celda luego de la demora inercial se informa de su presencia la cruce. La segunda regla representa la llegada de la cabeza del camión a la celda origen, pero el cruce no es informado hasta que se cumpla la demora inercial correspondiente. La tercera regla representa la llegada de las partes posteriores del camión que no alteran el estado de la celda pues el camión ya ha ingresado al cruce y la cola se va borrando del tramo.

Las celdas de la columna k-1, sólo permiten que los camiones se muevan hacia el cruce; eliminando los movimientos de ellos hacia los carriles adyacentes. Es decir, las celdas de la columna k-1 sólo pueden recibir un camión del vecino (0,-1).

Para las celdas de la columna k-1, de las reglas para los autos que les corresponden según el carril donde se encuentran, se modifican las que representan la llegada a la celda origen, es decir: Llega_Auto_Desde_Atrás, Llega_Auto_Desde_CarrilIz y Llega_Auto_Desde_CarrilDer. Para ellas sólo cambia la actualización de los ports externos, que en vez de enviar el estado de la celda, envían siempre 0 (vacía). Pues la regla Sale_Auto_Hacia_Cruce se encarga de enviar la presencia de un auto que ingresará al cruce y se especifica como:

Nuevo estado	Estado Vecindario	Delay	d	Nuevo Estado de los Ports
0	(0,0) = 1 and portvalue(x-c-haylugar) = 0	inercial	Conversión_Demora(velocidad(max))	send(1,y-c-hayvehículo)

Luego las demás reglas para estas celdas que representan la salida de autos se eliminan (Sale_Auto_Hacia_*).

Las celdas de la columna k-2 tienen las mismas reglas para los autos que les corresponden según el carril que ocupan pero las reglas para los camiones deben cambiar pues no tienen el vecindario completo, no tienen el vecino (0,2) y además su vecino (0,1) tiene comportamiento especial pues es el que compacta camiones (columna k-1). Por esto la macro EsCabeza es cambiada por CamiónCompleto, pues si la celda origen percibe hacia atrás todo el camión completo significa que es la cabeza del mismo. Luego el avance derecho de la cola del camión hacia (0,1) cambia de la siguiente forma:

Nuevo estado	Estado Vecindario	Delay	d
0	HayCamión(0,0) and (1,1) != (0,0) and (-1,1) != (0,0) and (0,1) = 0 and not(CamiónCompleto(0,0))	transport	0

Esta regla es la que hace que la cola del camión siga avanzando a pesar de no encontrar la cabeza pues ella fue enviada al cruce. La celda (0,1) va haciendo desaparecer cada una de estas partes del camión.

2.3.2. Cruces con circulación de camiones

El conjunto de los cruces se obtiene a partir de los tramos definidos como:

$$\text{CrucesCamiones} = \{ (c, maxc) / maxc \in N \wedge \exists\, t, t' \in (\text{TramosCamiones } U \text{ Tramos}) \wedge t = (p1, p2, n, a, dir, max) \wedge t' = (p1', p2', n', a', dir', max') \wedge t \neq t' \wedge (p1 = c \vee p2 = c) \wedge (p1' = c \vee p2' = c) \}$$

Los elementos de este conjunto se definen como puntos del espacio de dos dimensiones que representan los lugares donde se cruzan 2 ó más tramos con circulación de camiones, asociados con su velocidad máxima permitida. Se construye a partir de los conjuntos Tramos y TramosCamiones, buscando aquellos extremos donde coincide más de una calle. Pero un cruce siempre debe tener por lo menos un tramo de ingreso y uno de salida, pues si esto no sucede en realidad no se trata de una intersección de calles sino del lugar donde nacen o terminan. Esta restricción se escribe como:

$$\forall\, (c, maxc) \in \text{CrucesCamiones}: \exists\, t, t' \in (\text{TramosCamiones } U \text{ Tramos}) \wedge t = (p1, p2, n, a, dir, max) \wedge t' = (p1', p2', n', a', dir', max') \wedge t \neq t' \wedge [(p1 = p1' = c \wedge dir \neq dir') \vee (p1 = p2' = c \wedge dir = dir') \vee (p2 = p2' = c \wedge dir \neq dir')]$$

De esta forma los vehículos que ingresan al cruce por algún tramo siempre tendrán por lo menos una salida disponible.

Estos cruces son distintos de los que sólo tienen circulación de autos pues no son de estado binario y además deben poder distinguir por donde pueden o no salir los camiones (tramos de salida que permitan camiones). Los cruces de estado binario definidos en la sección 1.3.2, se utilizan cuando todos los tramos que intersecta son de estado binario (sin camiones). En cambio, si existe por lo menos un tramo t que lo tiene como extremo y $t \in$ TramosCamiones entonces el cruce se define como un elemento del conjunto CrucesCamiones. Para separar ambos casos se establece la siguiente restricción:

$$\forall\, (c, maxc) \in \text{CrucesCamiones}: \exists\, t \in \text{TramosCamiones} \wedge t = (p1, p2, n, a, dir, max) \wedge (p1 = c \vee p2 = c)$$

$$\forall\, (c, maxc) \in \text{Cruces}: \text{not} (\exists\, t \in \text{TramosCamiones} \wedge t = (p1, p2, n, a, dir, max') \wedge (p1 = c \vee p2 = c))$$

Cada cruce $(c, maxc) \in$ CrucesCamiones, se modela con un espacio Cell_DEVS de una dimensión con demora de transporte y bordes conectados. Donde cada celda se define como:

$$C_{0j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

con

I = $< \eta, P^x, P^y >$, donde

$\eta = 3$;

P^x = { (X_1, natural), (X_2, natural), (X_3, natural) };
P^y = { (Y_1, natural), (Y_2, natural), (Y_3, natural) }.

X = ***Y*** = N;

S:

$$s = \begin{cases} 1 & \text{si hay un auto en la celda;} \\ k : k \equiv r \bmod 10 \wedge 2 \leq r \leq 5 & \text{si hay un camión en la celda;} \\ 0 & \text{si la celda está vacía.} \end{cases}$$

N = { (0,-1), (0,0), (0,1) };

delay = transport;

d = Conversión_Demora(velocidad(maxc));

λ, δ_{int} y δ_{ext} se comportan como las funciones definidas por el formalismo Cell-DEVS para demoras de transporte.

La definición de la función **τ** se realizará luego de introducir el modelo acoplado porque todas las celdas utilizan la información de los ports para calcular su nuevo estado.

El modelo acoplado correspondiente al cruce (c,maxc) se define como:

CruceCamiones(k, In, Out, Out_SoloAutos, maxc) = < Xlist, Ylist, I, *X*, *Y*, n, {t_1,...,t_n}, η, N, C, B, Z, select >

donde,

Ylist = { (0,i) / $0 \leq i < k$}

Xlist = { (0,i) / $0 \leq i < k$}

I = <P^x, P^y>

P^x = {<$X_{\eta+1}$(0,i), binario> / $0 \leq i < k$ }
P^y = {<$Y_{\eta+1}$(0,i), binario> / $0 \leq i < k$ }

Estos ports serán denotados de la siguiente forma:

Port	Nombre	Comentario
$X_{\eta+1}$(0,i), $i \in$ In	x-t-hayvehículo	Este port se usa para saber si en el tramo existe un vehículo que desea ingresar al cruce (portvalue(x-t-hayvehículo) = 1).
$X_{\eta+1}$(0,i), $i \in$ Out	x-t-haylugar	Este port se usa para saber si en el tramo existe lugar para que salga un vehículo del cruce (portvalue(x-t-haylugar) = 0).
$Y_{\eta+1}$(0,i), $i \in$ In	y-t-haylugar	Este port informa al tramo si hay suficiente lugar en el cruce para el avance un vehículo.
$Y_{\eta+1}$(0,i), $i \in$ Out	y-t-hayvehículo	Este port informa al tramo si hay

		un vehículo que sale desde el cruce hacia él.

$\boldsymbol{X}$ = N

$\boldsymbol{Y}$ = N

n = 1

$\mathbf{t_1}$ = k

N = { (0,-1), (0,0), (0,1) }

C = { C_{ij} / i = 0 $\wedge$ j $\in$ [0, k-1] }, donde cada C_{ij} es un componente Cell-DEVS con demora y la especificación de cada celda ha sido descripta antes de introducir el modelo acoplado.

B = {$\varnothing$}

Z se construye siguiendo la definición dada por el formalismo Cell_DEVS, instanciada con este vecindario.

select = { (0,1), (0,0), (0,-1) }

Cruce(k, In, Out, Out_SoloAutos, maxc) significa que es un Cruce de longitud k (celdas) con velocidad máxima permitida de maxc (Km/h), donde las posiciones de In actúan como entradas hacia el cruce, las de Out son salidas del mismo para autos y camiones, y las de Out_SoloAutos son salidas exclusivas para autos (en el tramo acoplado no se permite la circulación de camiones). Los conjuntos In, Out y Out_SoloAutos se obtienen utilizando la función Ports_In_Out,

{I, O} = Ports_In_Out((c, maxc), Tramos U TramosCamiones)

Para establecer cuáles son las celdas de entrada y cuáles las de salida, se parte de la especificación de los tramos que lo tienen como extremo y se chequea el sentido de circulación de los vehículos en ellos. En el Apéndice A se muestra la función Ports_In_Out que realiza dicha tarea y devuelve dos conjuntos, el primero contiene cada tramo de ingreso al cruce que está conectado al mismo con el índice de la primera celda del cruce con la que se acopla; y el otro contiene la misma información pero para los tramos de salida. Esta función establece un ordenamiento de los tramos que se acoplan al mismo cruce, de forma tal que los tramos más cercanos entre sí ocupen posiciones vecinas, esto se logra ordenándolos según el ángulo de inclinación respecto a la recta y = y1 con c = (x1, y1). Este orden también es necesario para realizar el acoplamiento de los modelos.

De los conjuntos I y O sólo se necesitan los índices de las celdas que son entradas y las que son salidas, y además diferenciar entre las que se acoplan a tramos con circulación de camiones y las conectadas a tramos exclusivos para autos. Esto es:

In = { i+j / $\exists$t = (c1, c2, n, a, dir, max) $\wedge$ (t,i) $\in$ I $\wedge$ j $\in$ [0, n-1] }

Out = { i+j / $\exists$t = (c1, c2, n, a, dir, max) $\wedge$ (t,i) $\in$ O $\wedge$ t $\in$ TramosCamiones $\wedge$ j $\in$ [0, n-1] }

Out_SoloAutos = { i+j / $\exists$t = (c1, c2, n, a, dir, max) $\wedge$ (t,i) $\in$ O $\wedge$ t $\in$ Tramos $\wedge$ j $\in$ [0, n-1] }

Para determinar la cantidad de celdas que tiene el cruce (c, maxc), basta sumar el número de carriles que tiene cada tramo asociado al cruce pues cada celda se acopla a un carril. Es decir,

$$k = \sum_{\substack{(t,i)\in(I\cup O)\wedge \\ t=(c1,c2,n,a,dir)}} n \quad = \#(\text{In}) + \#(\text{Out}) + \#(\text{Out\_SoloAutos})$$

Para definir τ se debe tener en cuenta que el comportamiento de las celdas difiere si se trata de celdas de ingreso o de salida del cruce, por lo tanto se define un τ distinto para cada caso, en cada una de las siguientes secciones.

2.3.2.1. Celdas de salida del cruce

Las celdas de salida del cruce tienen diferente comportamiento de acuerdo al tipo de tramo (con o sin circulación de camiones) con que se acoplan.

Las celdas de salida del cruce que permiten que autos y camiones lo abandonen son:

$\{ (0,i) / i \in Out \}$

La función τ para estas celdas se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports
1	(0,0) = 0 and (0,-1) = 1 and [portvalue(x-t-haylugar) = 1 or EsCamión(portvalue(x-t-haylugar)) or (portvalue(x-t-haylugar) = 0 and random < p_{salir})] /* Llega auto que permanecerá dentro del cruce */	send(0,y-t-hayvehículo)
(0,-1)	(0,0) = 0 and HayCamión(0,-1) and [portvalue(x-t-haylugar) = 1 or EsCamión(portvalue(x-t-haylugar)) or (portvalue(x-t-haylugar) = 0 and random < p_{salir})] /* Llega camión que permanecerá dentro del cruce*/	send(0,y-t-hayvehículo)
0	(0,0) = 0 and (0,-1) = 1 and portvalue(x-t-haylugar) = 0 and random ≥ p_{salir} /* Llega auto que abandonará el cruce */	send(1,y-t-hayvehículo)
0	(0,0) = 0 and HayCamión(0,-1) and portvalue(x-t-haylugar) = 0 and random ≥ p_{salir} /* Llega camión que abandonará el cruce */	send((0,-1), y-t-hayvehículo)
0	(0,0) = 1 and (0,1) = 0	send(0,y-t-hayvehículo)
0	HayCamión(0,0) and (0,1) = 0	send(0,y-t-hayvehículo)
(0,0)	t /* en cualquier otro caso conserva el estado */	send(0,y-t-hayvehículo)

Aquí, p_{salir} es una constante que representa la probabilidad de que un vehículo que atraviesa la celda origen abandone el cruce en su próximo movimiento.

Estas reglas permiten que los autos o los camiones abandonen el cruce siempre y cuando haya lugar suficiente en el tramo y el valor aleatorio sea mayor que p_{salir}.

Las celdas de salida del cruce que sólo permiten que los autos lo abandonen, los camiones permanecen girando siempre al atravesarlas, son:

$\{ (0,i) / i \in Out\_SoloAutos \}$

La función τ para estas celdas se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports

1	(0,0) = 0 and (0,-1) = 1 and (portvalue(x-t-haylugar) = 1 or (portvalue(x-t-haylugar) = 0 and random < p_{salir})) /* Llega auto que permanecerá dentro del cruce */	send(0,y-t-hayvehículo)
(0,-1)	(0,0) = 0 and HayCamión(0,-1) /* Llega camión que siempre permanece dentro del cruce */	send(0,y-t-hayvehículo)
0	(0,0) = 0 and (0,-1) = 1 and portvalue(x-t-haylugar) = 0 and random ≥ p_{salir} /* Llega auto que abandonará el cruce */	send(1,y-t-hayvehículo)
0	(0,0) = 1 and (0,1) = 0	send(0,y-t-hayvehículo)
0	HayCamión(0,0) and (0,1) = 0	send(0,y-t-hayvehículo)
(0,0)	t /* en cualquier otro caso conserva el estado */	send(0,y-t-hayvehículo)

Estas reglas permiten que sólo los autos abandonen el cruce siempre y cuando haya lugar suficiente en el tramo y el valor aleatorio sea mayor que p_{salir}. Cuando llega un camión, éste siempre permanece dentro del cruce y sólo avanza si el vecino de adelante está vacío ((0,1) = 0), es decir nunca puede abandonar el cruce desde estas celdas.

2.3.2.2. Celdas de ingreso al cruce

Las celdas de ingreso al cruce son:

$\{ (0,i) / i \in In \}$

La función τ para estas celdas se define como:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports
1	(0,0) = 0 and (0,-1) = 1	send(1, y-t-haylugar)
1	(0,0) = 0 and portvalue(x-t-hayvehículo) = 1 and (0,-1) = 0	send(1, y-t-haylugar)
(0,-1)	(0,0) = 0 and HayCamión(0,-1)	send(1, y-t-haylugar)
portvalue(x-t-hayvehículo)	(0,0) = 0 and EsCamión(portvalue(x-t-hayvehículo)) and (0,-1) = 0	send(1, y-t-haylugar)
0	(0,0) = 1 and (0,1) = 0 and (0,-1) = 0 /* No hay vehículo con prioridad dentro del cruce*/	send(0, y-t-haylugar)
0	HayCamión(0,0) and (0,1) = 0 and (0,-1) = 0 /* No hay vehículo con prioridad dentro del cruce*/	send(0, y-t-haylugar)
0	(0,0) = 1 and (0,1) = 0 and ((0,-1) = 1 or HayCamión(0,-1)) /* Hay un vehículo con prioridad dentro del cruce*/	send(1, y-t-haylugar)
0	HayCamión(0,0) and (0,1) = 0 and ((0,-1) = 1 or HayCamión(0,-1)) /* Hay un vehículo con prioridad dentro del cruce*/	send(1, y-t-haylugar)
(0,0)	t /* En cualquier otro caso conserva estado */	-

Para estas reglas es necesario establecer la actualización de los ports, pues no siempre se envía el nuevo estado. En particular, en las reglas 3 y 4 el nuevo estado representa la presencia de un camión, y puede pasar que el tramo acoplado sea de estados binarios y no reconozca este valor;

por eso se envía el valor 1 que representaría que la celda está ocupada. Con esto le alcanza al tramo para decidir que debe esperar a que se vacíe y en realidad no necesita conocer con mayor detalle si se trata de un auto o camión. Luego sólo se envía 0 por el port de salida sólo cuando la celda origen está vacía y además no existe en el cruce un vehículo que quiera avanzar hacia ella. Esto se modela así para que los vehículos que quieran ingresar al cruce no lo puedan hacer cuando existe otro en él que quiera avanzar a la misma posición ("prioridad de los vehículos dentro del cruce").

2.3.3. Acoplamiento entre Tramos y Cruces

El acoplamiento de estos modelos representa el movimiento de vehículos que ingresan al (salen del) cruce desde (hacia) algún tramo y es similar al que fuera descripto para las calles exclusivas para autos. Un cruce c = (p, maxc) tiene influencias sobre los tramos t que lo tienen como un extremo, es decir, el conjunto de modelos que influye el Cell-DEVS de c es el siguiente:

$I_c = \{ M_t / t \in (Tramos\ U\ TramosCamiones) \wedge t = (p1, p2, n, a, dir, max) \wedge (p1 = p\ OR\ p2 = p) \}$

Luego un tramo t tiene definidas las influencias sobre los modelos de los dos cruces que tienen como extremo, es decir

$$I_t = \{M_{c1}\}\ U\ \{M_{c2}\}, \text{ si } t = (p1, p2, n, a, dir, max) \text{ y } (\exists\ v1, v2 \in N : c1, c2 \in (Cruces\ U\ CrucesCamiones) \wedge c1 = (p1, v1) \wedge c2 = (p2, v2))$$

Estos ports se acoplan definiendo la función Z. Para establecer el índice de celda del cruce y la del tramo que tienen la conexión se utiliza la función Ports_In_Out (definida en el apéndice A) que realiza dicha tarea y devuelve dos conjuntos, el primero contiene cada tramo de ingreso al cruce que está conectado al mismo con el índice de la primera celda del cruce con la que se acopla; y el otro contiene la misma información pero para los tramos de salida. Esta función establece un ordenamiento de los tramos que se acoplan al mismo cruce, de forma tal que los tramos más cercanos entre sí ocupen posiciones vecinas dentro del mismo, esto se logra ordenándolos según el ángulo de inclinación respecto a la recta y = y1 con p = (x1,y1). Utilizando Ports_In_Out se obtienen los conjuntos I y O de la siguiente manera:

$$\{I, O\} = Ports\_In\_Out(c, Tramos\ U\ TramosCamiones)$$

Para cada (t,i) $\in$ I con t = (p1, p2, n, a, dir, max) se debe conocer la cantidad de celdas del tramo porque el acoplamiento se realiza en la primera (0) y última celda de cada carril (k-1, si k es la cantidad de celdas del tramo t). Esto se obtiene calculando la longitud del tramo (a partir de p1 y p2), y luego dividiéndola por el tamaño definido para la celda (en el Apéndice A se encuentra definida la función Ctd_Celdas). De esta forma la cantidad de celdas que tendrá t se calcula como:

$$k = Ctd\_Celdas(t)$$

Z se define como:

$$Z_{tc} : Y_{\eta+1}(j, k-1)_t \rightarrow X_{\eta+1}(0,i+j)_c, \forall (j \in N, j \in [0, n-1])$$
$$Z_{ct} : Y_{\eta+1}(0,i+j)_c \rightarrow X_{\eta+1}(j, k-1)_t, \forall (j \in N, j \in [0, n-1])$$

Para cada (t,i) $\in$ O con t = (p1, p2, n, a, dir, max), Z se define como:

$$Z_{ct} : Y_{\eta+1}(0,j+i)_c \rightarrow X_{\eta+1}(n-1-j, 0)_t, \forall (j \in N, j \in [0, n-1])$$
$$Z_{tc} : Y_{\eta+1}(n-1-j, 0)_t \rightarrow X_{\eta+1}(0,j+i)_c, \forall (j \in N, j \in [0, n-1])$$

2.4. Choques

Para especificar los choques se debe indicar sobre qué tramos se pueden producir, así se construye el siguiente conjunto:

$$TramosChoques = \{ t / t \in TramosCamiones \}$$

Este conjunto contiene tramos en los que en algún momento de la simulación, en forma no determinística, se pueden producir los efectos de un choque. Esto es que en algunos sectores del tramo no se permita la circulación de vehículos durante un cierto tiempo. En los tramos con estado binario no podrán definirse los choques porque éstos se han definido utilizando más valores posibles para las variables, de ahí que los tramos deben pertenecer al conjunto TramosCamiones y no al de Tramos.

Un choque se modela como la generación espontánea de un sector del tramo, con forma de rombo y tamaño fijo, que no puede ser atravesado por vehículos. El choque se inicia en una celda que cambiará su estado a 6, y luego se expande a las vecinas (1,0), (0,-1), (0,1) y (-1,0) que pasarán a estado 7, como se ilustra en la Figura 47. La utilización de 2 valores distintos (6 y 7) surge para poder diferenciar a la celda donde se origina el choque del resto de las afectadas. Sólo las celdas que están en contacto directo con la de estado 6, pasarán a estado de choque (s = 7); entonces las adyacentes a las celdas de estado 7 no continúan expandiendo el choque en forma indefinida.

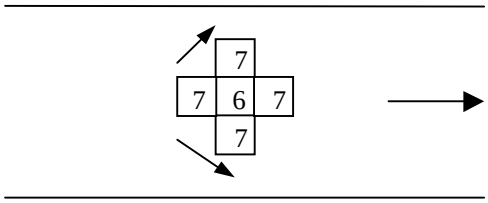


Figura 47 – Tramo con un choque

En esta figura las flechas indican el sentido de circulación de los vehículos en el tramo y cómo se deben desviar cuando se acercan al sector donde se ha producido el choque.

La regla que origina el choque se define como:

Nuevo Estado	Estado del vecindario	d	Delay	Nombre de la regla
6	(0,0) = 1 and random > p_{choque} and NoHayCamiónCerca	Conversión_Demora(velocidad(max))	transport	Genera_Choque

Esta regla representa la generación de un choque en la celda origen, que sólo se produce cuando existe un vehículo en ella, no hay ningún camión en su entorno y la función random devuelve un valor más grande que la constante p_{choque}. La condición random > p_{choque} se debe hacer verdadera de acuerdo a la frecuencia de choques que existe en el tramo, que se puede obtener de datos estadísticos o del log de la simulación como se describe en la sección 1.4.7. Cuando random ≤ p_{choque} se utilizan las reglas definidas para la circulación normal de vehículos (autos y camiones). En esta regla se utiliza la macro NoHayCamiónCerca que se define como:

NoHayCamiónCerca ≡ not(HayCamión(0,-1) or HayCamión(0,1) or HayCamión(1,-1) or HayCamión(1,0) or HayCamión(1,1) or HayCamión(-1,-1) or HayCamión(-1,0) or HayCamión(-1,1))

Esta macro verifica que ninguno de los vecinos de la celda origen contenga un camión.

Para que se produzca un choque se pide que no haya ningún camión en el entorno de la celda porque si lo hubiera podría ocurrir que parte del mismo sea afectado por el choque y parte no. Por lo tanto no quedaría el camión completo circulando y se generarían inconsistencias en las reglas de movimiento. Esto se puede solucionar agregando reglas que cuando ocurre esto se encarguen de eliminar el camión completo o restringiendo la generación de choques para que no afecten a los camiones que se encuentran en las celdas. Esta segunda opción, que es la que ha sido modelada aquí, sólo restringe la generación del choque donde no hay camiones, pero si éstos se aproximan luego del accidente deberán desviar su rumbo para esquivarlo. De esta forma se logra una solución más simple y que permite modelar la formación de congestión en torno al choque, que es el objetivo de esta sección.

Además se define la regla que expande el choque a las celdas vecinas y genera el rombo donde no se permite la circulación de vehículos. Aquí:

Nuevo Estado	Estado del vecindario	d	Delay	Nombre de la regla
7	(0,0) != 7 and ((-1,0) = 6 or (1,0) = 6 or (0,-1) = 6 or (0,1) = 6)	0	transport	Expande_Choque

Esta regla representa que la celda origen detecta la generación del choque en una de sus vecinas y por consiguiente en forma instantánea pasa a estado 7, que indica que los vehículos no la pueden atravesar por estar afectada por el choque. Así por la condición impuesta en esta regla sólo las celdas del rombo en el entorno de la posición con estado 6, cambiarán su estado a 7. Además esta regla representa que las celdas afectadas por el choque se mantendrán en estado 7 mientras posición donde se originó conserve el estado 6.

Con las reglas Genera_Choque y Expande_Choque puede ocurrir que se eliminen algunos vehículos de la simulación, esto representaría los autos chocados que dejan de circular.

Por último queda definir las reglas que hacen que desaparezca el estado de choque de las celdas, luego de un cierto tiempo en el cual se ha liberado la calle de los vehículos chocados.

Nuevo Estado	Estado del vecindario	d	Delay	Nombre de la regla
0	(0,0) = 6 or (0,0) = 7	d_{choque}	transport	Desaparece_Choque

Esta regla hace que la celda origen pase a estar vacía y recupere los movimientos normales de los vehículos, que ahora la pueden volver a atravesar. Esto ocurre luego de que la celda tenga estado de choque durante el tiempo representado por la constante d_{choque}.

Estas reglas se deben incorporar a las definidas para el movimiento de los vehículos en los tramos del conjunto TramosChoques, que serán aplicadas las reglas en el siguiente orden:

1. Genera_Choque
2. Expande_Choque
3. Desaparece_Choque
4. Reglas de movimiento de autos y camiones

Donde las últimas reglas son aquellas definidas para los tramos de la sección 2.3.1 para la circulación de vehículos.

Convención 5

Las reglas del comportamiento de los modelos de esta sección se deben evaluar en el orden en que fueron definidas. Así, en caso de que más de 2 reglas puedan ser aplicadas, la primera que se encuentre será la que establezca el nuevo estado de la celda. En algunos casos todas las reglas que pueden ser potencialmente aplicadas llevan al mismo estado de la celda, pero en otras no y de ahí la importancia de aclarar en qué orden se deben evaluar.

Las reglas definidas para el movimiento de autos y camiones en los tramos (sección 2.3.1) no se modifican por la presencia de choques (salvo aquellas definidas para las celdas borde). Esto se debe a que establecen como condición para poder avanzar que la celda se encuentre vacía (estado 0), de esta forma ningún vehículo intentará moverse hacia una celda afectada por un choque pues ésta tiene estado 6 ó 7. Para celdas borde de los tramos se deben adaptar las reglas Genera_Choque y Expande_Choque de acuerdo al vecindario de cada una. Esto es, eliminar de esas reglas las condiciones que se piden sobre los vecinos que no existen. Para los carriles de los tramos que no tienen el vecindario de 9x9 alrededor de la celda origen también deben ser adaptadas las reglas Genera_Choque y Expande_Choque, de acuerdo a su conformación. Esto es, eliminar de esas reglas las condiciones que se piden sobre los vecinos que no existen para estas celdas.

Los choques siempre se originan en los tramos pero se pueden expandir hacia los cruces acoplados cuando afectan a las celdas del borde, como se puede ver en la Figura 48.

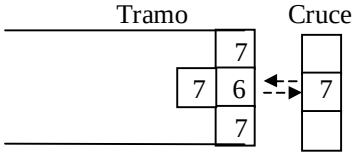


Figura 48 – Choque que afecta el Cruce

Para modelar la expansión de los choques en los cruces se definen nuevas reglas y se modifican algunas de las definidas. Los choques sólo pueden afectar a los cruces del conjunto CrucesCamiones, pues sólo se generan en los tramos de estado no binario.

Para las celdas de salida del cruce se agregan las siguientes reglas:

Nuevo Estado	Estado del vecindario	d	Delay	Nuevo Estado de los Ports
7	portvalue(x-t-haylugar) = 6	0	transport	send(7, y-t-hayvehículo)
0	(0,0) = 7	d_{choque}	transport	send(0, y-t-hayvehículo)

Estas 2 reglas representan la expansión del choque hacia el cruce y la desaparición del mismo luego de tiempo representado por la constante d_{choque}.

Para las celdas de ingreso al cruce se agregan algunas reglas y se modifican otras. Se debe tener en cuenta que las reglas de estas celdas codifican su estado de ellas y el de la posición de atrás cuando realizan la actualización de los ports externos. Por esto, se debe contemplar en estas reglas el caso en que se expande el choque a la celda origen y cuando se expande a la vecina de atrás (celda (0,-1)). Para modelar el primer caso se agregan 2 reglas:

Nuevo Estado	Estado del vecindario	d	Delay	Nuevo Estado de los Ports

7	portvalue(x-t-hayvehículo) = 6	0	transport	send(7, y-t-haylugar)
0	(0,0) = 7	d_{choque}	transport	send(0, y-t-haylugar)

Para el caso que la celda de atrás sea la que tiene estado de choque se deben cambiar las reglas:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports
1	(0,0) = 0 and portvalue(x-t-hayvehículo) = 1 and (0,-1) = 0	send(1, y-t-haylugar)
portvalue(x-t-hayvehículo)	(0,0) = 0 and EsCamión(portvalue(x-t-hayvehículo)) and (0,-1) = 0	send(1, y-t-haylugar)
0	(0,0) = 1 and (0,1) = 0 and (0,-1) = 0 /* No hay vehículo con prioridad dentro del cruce*/	send(0, y-t-haylugar)
0	HayCamión(0,0) and (0,1) = 0 and (0,-1) = 0 /* No hay vehículo con prioridad dentro del cruce*/	send(0, y-t-haylugar)

por:

Nuevo Estado	Estado del vecindario	Nuevo Estado de los Ports
1	(0,0) = 0 and portvalue(x-t-hayvehículo) = 1 and ((0,-1) = 0 or (0,-1) = 7)	send(1, y-t-haylugar)
portvalue(x-t-hayvehículo)	(0,0) = 0 and EsCamión(portvalue(x-t-hayvehículo)) and ((0,-1) = 0 or (0,-1) = 7)	send(1, y-t-haylugar)
0	(0,0) = 1 and (0,1) = 0 and ((0,-1) = 0 or (0,-1) = 7) /* No hay vehículo con prioridad dentro del cruce*/	send(0, y-t-haylugar)
0	HayCamión(0,0) and (0,1) = 0 and ((0,-1) = 0 or (0,-1) = 7) /* No hay vehículo con prioridad dentro del cruce*/	send(0, y-t-haylugar)

La única modificación realizada en estas reglas es agregar la condición (0,-1) = 7, pues en estos casos es equivalente que la celda (0,-1) esté vacía o afectada por el choque. Esto se debe a que se está verificando que en esa celda no exista un vehículo que quiera avanzar hacia la celda origen.

3. Apéndices

Aquí se resumen algunos aspectos importantes que se encuentran distribuidos en varias secciones del trabajo y se definen otros con el objetivo de simplificar la explicación de los modelos.

3.3. Apéndice A

En esta sección se define un conjunto de funciones que se utilizan fundamentalmente en el mapeo entre el lenguaje de especificación de secciones de ciudad y los modelos Cell-DEVS. Entre ellas la de mayor importancia es Ports_In_Out que es la que modela cómo se efectúa el acoplamiento entre tramos y cruces. Luego la sección 3.3.2 presenta un resumen de las funciones aleatorias y constantes que se utilizan principalmente en las reglas de especificación del comportamiento de los modelos.

3.3.1. Funciones Auxiliares

Ctd_Celdas (t: Tramos): N
Esta función establece la cantidad de celdas que contendrá el tramo t; calculando su longitud (a partir de la distancia entre sus extremos), y luego dividiéndola por el tamaño definido para la celda.

Sea t = (c1, c2, n, a, dir, max) untramo
Si a = 0 entonces ℓ = Long_Recta(c1,c2) /* Long_Recta se define más adelante */
Sino ℓ = Perímetro(c1,c2) /* Perímetro se define más adelante */
k = $\lceil \ell$ / long_celda $\rceil$
devolver (k)
Fin Ctd_Celdas.

Ctd_Celdas_Cruce (c: Cruces, T: Π(Tramos)): N
Esta función establece el número de celdas que contendrá el cruce c; sumando la cantidad de carriles que contiene cada tramo acoplado. Pues en cada cruce existirá una celda por cada carril de cada tramo acoplado.

Sea c = (p, maxc)

$$k = \sum_{\substack{t \in T \wedge t=(c1,c2,n,a,dir) \\ \wedge (c1=p \vee c2=p)}} n$$

devolver (k)
Fin Ctd_Celdas_Cruce.

Long_Recta ((x1,y1): Puntos, (x2,y2): Puntos): P
Esta función calcula la distancia entre 2 puntos del espacio cartesiano.

$$l = \sqrt{|x1 - x2|^2 + |y1 - y2|^2}$$

devolver ($|\ell|$)
Fin Long_Recta.

Perímetro ((x1,y1): Puntos, (x2,y2): Puntos): P
Esta función calcula el perímetro de una las semicircunferencias, cuyo diámetro es el segmento que une los puntos (x1,y1) y (x2,y2).

r = Long_Recta ((x1,y1), (x2,y2)) / 2
/* r es el radio de la circunferencia */
p = (2πr) / 2
/* se devuelve la mitad del perímetro total de la circunferencia */
devolver (p)
Fin Perímetro.

Inclinación ((x1,y1): Puntos, (x2,y2): Puntos): ángulo
Esta función calcula el ángulo de inclinación del segmento determinado por los puntos (x1,y1) y (x2,y2), respecto de la recta y = y1. Esto se logra despejando α de:
sen(α) = (cat-op / hipotenusa) = |y1-y2|/Long_Recta((x1,y1), (x2,y2))
Entonces:

$$\alpha = \text{arcsen}\left(\frac{|y1 - y2|}{Long\_Recta((x1, y1), (x2, y2))} \right)$$

Si [(y1 > y2) ∨ (y1 = y2 ∧ x1 < x2)] entonces α = α + π

devolver (α)

Cabe destacar que los parámetros de entrada deben ser distintos, pues si no es así la longitud de la recta por la que se divide es cero y se indefine el cociente. Por otro lado, hay que tener en cuenta que el segmento determinado por (x1,y1) y (x2,y2), tendrá una inclinación distinta para cada uno de sus extremos; como se muestra en la Figura 49. Para obtener el ángulo respecto a alguno de ellos, éste debe ser pasado como primer parámetro en esta función (Inclinación).

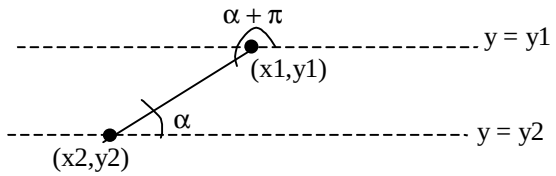


Figura 49 – Ángulos de inclinación

Fin Inclinación.

Ports_In_Out((c, maxc): Cruce, T: cjto de Tramos): { (t,i) / t ∈ Tramos, i ∈ N }, { (t,i) / t ∈ Tramos, i ∈ N}
Esta función recibe como parámetros un cruce (c, maxc) y el conjunto de tramos T, y devuelve dos conjuntos. El primero contendrá los tramos que llegan al cruce (con dirección de circulación de vehículos hacia c) y la posición (i) en que el primer carril se conecta al mismo. El segundo contendrá los tramos que salen desde cruce (con dirección de circulación opuesta a c) y la posición (i) en que el primer carril se conecta al mismo.

Para establecer la posición en que cada tramo se acopla al cruce se define un ordenamiento según el ángulo de inclinación de los mismos respecto a la recta y = y1 con c = (x1,y1). Así, a medida que los tramos tengan mayor ángulo les corresponderán las posiciones mayores dentro del cruce. Logrando que los tramos cercanos ocupen las celdas adyacentes dentro del cruce.

Por la restricción impuesta en el lenguaje de especificación a lo sumo puede haber dos tramos que se acoplan al mismo cruce con igual inclinación pero tendrán distinta dirección.

Para establecer un ordenamiento total de los tramos, en el caso de tener igual ángulo se adopta la convención de primero acoplar el tramo con dirección hacia c.

Para construir los conjuntos que devuelve esta función, primero se obtienen los tramos del conjunto T que tienen como algún extremo a c:

T' = { t / t ∈ T ∧ t = (c1,c2, n, a, dir, max) ∧ (c1 = c ∨ c2 = c) }

Luego se evalúan los ángulos de inclinación de estos tramos y en forma creciente se les asignan las posiciones del cruce por las que se acoplarán. Esta información se guarda en el conjunto V que contiene tuplas de la forma (t,i,α), que representan que el primer carril del tramo t se conecta al cruce (c, maxc) por la celda i y que α es el ángulo de inclinación de t. Este conjunto se describe como:

$$V = \left\{ \begin{array}{l} ((c_1, c_2, n, a, dir, max), i, \alpha) / (c_1, c_2, n, a, dir, max) \in T' \wedge i \text{ se define en (1)} \wedge \\ \qquad [(c \neq c_1 \wedge \alpha = Inclinación(c, c_1)) \vee (c \neq c_2 \wedge \alpha = Inclinación(c, c_2))] \end{array} \right\}$$

$$(1)\ i = \left\{ \begin{array}{ll} \sum_{(c_1',c_2',n',a',dir',max') \in T' \wedge Inclinación(c_1',c_2') < \alpha} n' & si \quad (dir = 1 \Rightarrow c_2 = c) \wedge (dir = 0 \Rightarrow c_1 = c) \\ \sum_{\substack{(c_1',c_2',n',a',dir',max') \in T' \wedge Inclinación(c_1',c_2') \leq \alpha \wedge \\ (c_1',c_2',n',a',dir',max') \neq (c_1,c_2,n,a,dir,max)}} n' & si \quad (dir = 0 \Rightarrow c_2 = c) \wedge (dir = 1 \Rightarrow c_1 = c) \end{array} \right.$$

Para calcular i se considera que si 2 tramos tienen igual inclinación, primero se conecta al cruce el tramo que se dirige hacia c, y en el primer caso $((dir = 1 \Rightarrow c_2 = c) \wedge (dir = 0 \Rightarrow c_1 = c))$ t tiene dirección hacia c. Entonces se suman la cantidad de carriles de los tramos con inclinación menor estricto que α, porque si hubiera otro tramo con igual inclinación seguro ocupa las posiciones posteriores a t.
En el segundo caso $((dir = 0 \Rightarrow c_2 = c) \wedge (dir = 1 \Rightarrow c_1 = c))$ t tiene dirección opuesta a c, entonces se suman la cantidad de carriles de los tramos con inclinación menor o igual que α porque si hubiera otro tramo con igual inclinación seguro ocupa las posiciones anteriores a t.

Finalmente, las tuplas de V son separadas en aquellas que corresponden a tramos de ingreso al cruce y las que corresponden a los de salida. Esto se logra chequeando la dirección de circulación de los vehículos definida para cada tramo. Así se construye el conjunto In con los tramos de entrada a (c, maxc) y el conjunto Out con los de salida.

In = { (t,i) / (t, i,α) ∈ V ∧ t = (c1,c2, n, a, dir, max) ∧ [(c1 = c ∧ dir = 0) ∨
(c2 = c ∧ dir = 1)] }

Out = { (t,i) / (t, i,α) ∈ V ∧ t = (c1,c2, n, a, dir, max) ∧ [(c1 = c ∧ dir = 1) ∨
(c2 = c ∧ dir = 0)] }

devolver (In, Out)

Fin Ports_In_Out.

Conversión_Demora(veloc: N): N, definida como:

$$\text{Conversión\_Demora(veloc)} = \frac{long\_celda(km) * (60)^2}{veloc(km/h)}$$

Esta función divide la distancia que se debe recorrer (la longitud de una celda) sobre la velocidad utilizada para hacerlo, obteniendo el tiempo necesario o demora. El factor de $(60)^2$ se utiliza para devolver la demora en segundos.

3.3.2. Constantes y Funciones Aleatorias

A lo largo del trabajo se utilizan varias funciones aleatorias y constantes que representan aspectos reales del problema, cuyo resumen y descripción se presenta a continuación.

long_celda (en km)
ancho_celda (en km)
La longitud de la celda representa la longitud del auto más la distancia entre autos cuando los vehículos se encuentran detenidos, como fuera utilizado para los modelos de autómatas celulares planteados en [W95], [WNW97] y [NSPLDB98], entre otros, asignando el valor de 0,0075 km (7,5 m).

d_{bache} (demora que corresponde a la presencia de bache)
d_{baden} (demora que corresponde a la presencia de badén)
$d_{Lomoburro}$ (demora que corresponde a la presencia de lomo de burro)
$d_{SeñalStop}$ (demora que corresponde a la presencia de señal de stop)
$d_{SeñalEscuela}$ (demora que corresponde a la presencia de señal de escuela)
$d_{bocacalles}$ (demora que corresponde a la presencia de bocacalles)
$d_{Serrucho}$ (demora que corresponde a la presencia de serrucho)
d_{tren} (demora que representa la velocidad del tren)

p_{salir} (probabilidad de que un vehículo salga del cruce)

velocidad: N → N, es una función aleatoria con la distribución probabilística característica del flujo de vehículos. Se espera que pocos autos circulen con las velocidades extremas (máxima y mínima) y que la mayoría lo haga con velocidades intermedias. Recibe como parámetro el valor máximo que puede alcanzar la función y en base a él se obtiene la media $\bar{x} = \frac{2}{3} * max(km/h)$ y desvío estándar $\sigma = \frac{1}{3} * max(km/h)$, en este caso max es el parámetro pasado a la función. El valor que retorna es un natural que representa una velocidad en km/h elegida en forma aleatoria de acuerdo a la distribución descripta.

estacionar (función aleatoria que elige un valor sobre un conjunto de demoras grandes)

veloc_camión: N → N, es una función aleatoria que recibe como parámetro el valor máximo que puede alcanzar la función y retorna un valor natural que representa una velocidad en km/h. Se utiliza para modelar la velocidad de los camiones.

3.4. Apéndice B - Descripción de las reglas de especificación de los tramos

A continuación se presentan las reglas básicas del movimiento de autos utilizadas para la definición de los tramos binarios; con los vecinos definidos según la siguiente figura:

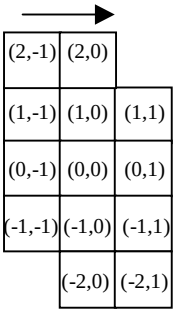


Figura 50 - Vecindario de la celda origen

Nuevo estado	Estado Vecindario	Nombre	Delay
1	(0,0) = 0 and (0,-1) = 1	Llega_Desde_Atrás	transport

Esta regla representa el movimiento de un auto que avanza desde atrás derecho hacia la celda origen. La demora se utiliza para modelar el tiempo que le lleva al vehículo llegar a la nueva posición.

Nuevo estado	Estado Vecindario	Nombre	Delay
1	(0,0) = 0 and (-1,-1) = 1 and (-1,0) = 1 and (0,-1) = 0	Llega_Desde_CarrilDer	transport

Esta regla representa el movimiento de un auto que avanza desde el carril de la derecha hacia la celda origen. Para que este movimiento sea posible se deben cumplir 3 condiciones, la primera es que la celda origen debe estar vacía ((0,0) = 0), la segunda que el auto no pueda avanzar derecho sobre su propio carril ((-1,-1) = 1 and (-1,0) = 1). Esto último se necesita pues si tuviera el camino libre sobre su carril entonces debe avanzar derecho, pues los autos siempre intentan primero ese movimiento. La tercera condición es que no haya otro auto que pueda avanzar a la celda origen derecho desde su carril ((0,-1) = 0), pues ese auto tiene prioridad sobre los que cambian de carril.

Nuevo estado	Estado Vecindario	Nombre	Delay
1	(0,0) = 0 and (1,-1) = 1 and (1,0) = 1 and (0,-1) = 0	Llega_Desde_CarrilIz_ConPrioridad	transport

Esta regla representa el movimiento de un auto que avanza desde el carril de la izquierda hacia la celda origen. Para que este movimiento sea posible se deben cumplir 4 condiciones, la primera es que la celda origen debe estar vacía ((0,0) = 0), la segunda que el auto no pueda avanzar derecho sobre su propio carril ((1,-1) = 1 and (1,0) = 1). Esto último se necesita pues si tuviera el camino libre sobre su carril entonces debe avanzar derecho, pues los autos siempre intentan primero ese movimiento. Luego de intentar el movimiento hacia adelante lo hará hacia la izquierda, por esto para que avance hacia la origen (a la derecha), se debe pedir también que no pueda avanzar hacia izquierda. En esta regla se asume que el vecindario impide que el auto pueda realizar ese movimiento, por lo que si no avanza derecho luego lo intenta hacia derecha. Por eso el nombre de la regla pues el movimiento hacia derecha tiene más prioridad que lo normal. La cuarta condición es que no haya otro auto que pueda avanzar a la celda origen derecho desde su carril ((0,-1) = 0), pues ese auto tiene prioridad sobre los que cambian de carril.

Nuevo estado	Estado Vecindario	Nombre	Delay
1	(0,0) = 0 and (1,-1) = 1 and (1,0) = 1 and (0,-1) = 0 and ((2,-1) = 1 or (2,0) = 1)	Llega_Desde_CarrilIz_SinPrioridad	transport

Esta regla representa el movimiento de un auto que avanza desde el carril de la izquierda hacia la celda origen. Para que este movimiento sea posible se deben cumplir 4 condiciones, la primera es que la celda origen debe estar vacía ((0,0) = 0), la segunda que el auto no pueda avanzar derecho sobre su propio carril ((1,-1) = 1 and (1,0) = 1). Esto último se necesita pues si tuviera el camino libre sobre su carril entonces debe avanzar derecho, pues los autos siempre intentan primero ese movimiento. Luego de intentar el movimiento hacia adelante lo hará hacia la izquierda, por esto para que avance hacia la origen (a la derecha), se debe pedir también que no pueda avanzar hacia izquierda ((2,-1) = 1 or (2,0) = 1). En esta regla se asume que el vecindario de la celda (1,-1) le permite realizar los 3 movimientos definidos (derecho, diagonal izquierda y diagonal derecha), por lo que si no avanza derecho luego lo intenta hacia izquierda y por último hacia derecha. Por eso el nombre de la regla pues el movimiento hacia derecha no tiene prioridad frente a los demás movimientos posibles. La cuarta condición es que no haya otro auto que pueda avanzar a la celda origen derecho desde su carril ((0,-1) = 0), pues ese auto tiene prioridad sobre los que cambian de carril.

Nuevo estado	Estado Vecindario	Nombre	Delay
1	(0,0) = 0 and portvalue(x-c-hayauto) = 1	Llega_Desde_Cruce	transport

Esta regla representa el movimiento de un auto que avanza desde el cruce acoplado al tramo hacia la celda origen. Para ello el estado del port debe ser 1 y la celda origen debe estar vacía. Es importante notar que el estado del port es 1 sólo cuando el vehículo saldrá del cruce.

Nuevo estado	Estado Vecindario	Nombre	Delay
0	(0,0) = 1 and (0,1) = 0	Sale_Hacia_Adelante	transport

Esta regla representa el movimiento de un auto que avanza desde la celda origen hacia la posición de adelante. Para que sea posible la posición de destino debe estar vacía ((0,1) = 0).

Nuevo estado	Estado Vecindario	Nombre	Delay
0	(0,0) = 1 and (-1,0) = 0 and (-1,1) = 0	Sale_Hacia_CarrilDer_ConPrioridad	transport

Esta regla representa el movimiento de un auto que avanza hacia el carril de la derecha desde la celda origen. Para que este movimiento sea posible se deben cumplir 4 condiciones, la celda origen debe estar ocupada por un auto ((0,0) = 1), la celda de destino debe estar vacía ((-1,1) = 0), en el carril derecho no puede haber otro auto que quiera acceder a la misma posición con mayor prioridad (avance recto), para esto se pide que (-1,0) = 0. Y por último, se debe pedir que no haya otro auto que con movimiento hacia izquierda (tiene prioridad sobre el que es hacia derecha) acceda a la misma posición. En esta regla se asume que el vecindario de la celda (-1,1) impide que un auto pueda realizar ese movimiento, por lo que si ninguno avanza derecho luego tienen prioridad los que lo hacen desde izquierda (desde celda origen). Por eso el nombre pues el movimiento hacia derecha tiene más prioridad que lo normal.

Nuevo estado	Estado Vecindario	Nombre	Delay
0	(0,0) = 1 and (-1,0) = 0 and (-1,1) = 0 and ((-2,1) = 0 or	Sale_Hacia_CarrilDer_SinPrioridad	transport

	(-2,0) = 0)		

Esta regla representa el movimiento de un auto que avanza hacia el carril de la derecha desde la celda origen. Para que este movimiento sea posible se deben cumplir 4 condiciones, la celda origen debe estar ocupada por un auto ((0,0) = 1), la celda de destino debe estar vacía ((-1,1) = 0), en el carril derecho no puede haber otro auto que quiera acceder a la misma posición con mayor prioridad (avance recto), para esto se pide que (-1,0) = 0. Y por último, se debe pedir que no haya otro auto que con movimiento hacia izquierda (tiene prioridad sobre el que es hacia derecha) acceda a la misma posición ((-2,1) = 0 or (-2,0) = 0). En esta regla se asume que el vecindario de la celda (-1,1) permite a los autos realizar ese movimiento, a diferencia de la regla anterior. Por esto el movimiento hacia el carril derecho desde la celda origen no tiene prioridad.

Nuevo estado	Estado Vecindario	Nombre	Delay
0	(0,0) = 1 and (1,0) = 0 and (1,1) = 0	Sale_Hacia_CarrilIz	transport

Esta regla representa el movimiento de un auto que avanza desde la celda origen hacia el carril de la derecha. Para que este movimiento sea posible se deben cumplir 3 condiciones, la celda origen debe estar ocupada ((0,0) = 1), la celda de destino debe estar vacía ((1,1) = 0) y en el carril izquierdo no puede haber otro auto que quiera acceder a la misma posición con mayor prioridad (avance recto), para esto se pide que (1,0) = 0.

Nuevo estado	Estado Vecindario	Nombre	Delay
0	((0,0) = 1 portvalue(x-c-haylugar) = 0)	Sale_Hacia_Cruce	inercial

Esta regla representa el movimiento de un auto que avanza desde la celda origen hacia el modelo acoplado del cruce. Para que un vehículo pueda ingresar al cruce deberá chequear la celda a la que quiere ingresar y la anterior a ésta, pues si un vehículo dentro del cruce quiere acceder a la misma posición éste tendrá prioridad. Por lo tanto, esta regla representa el avance del vehículo desde la celda origen hacia la celda acoplada del cruce, siempre y cuando ésta y su anterior se encuentren vacías (portvalue(x-c-haylugar) = 0). Para ingresar al cruce, es decir abandonar la celda origen, se modela con demora inercial porque representa el comportamiento de los vehículos al llegar a las esquinas y en general este tipo de maniobra se realiza con más precaución que el avance sobre el resto de la calle. Los automotores avanzan sobre el cruce si durante un cierto tiempo (demora inercial) tienen el camino libre, caso contrario esperan a que pasen los vehículos que ya se encuentran en el mismo. De esta forma los vehículos que circulan dentro del cruce tienen mayor probabilidad de avanzar que aquellos que desean ingresar y estos últimos no avanzan si no tienen el camino libre por el tiempo que les lleva realizar la maniobra.

Nuevo estado	Estado Vecindario	Nombre	Delay
(0,0)	t	Default	transport

Por defecto la celda conserva su estado. Las reglas de los modelos fueron planteadas en función de las condiciones que provocan cambios de estado en las celdas, por lo que se agrega esta regla para todos los demás casos que no fueron especificados explícitamente.

Los tramos con vehículos que estacionan sobre los carriles del borde, se especifican como:
AutosEstacionados = { (t, n1) / t ∈ Tramos ∧ n1 ∈ { 0, 1 } ∧ t = (c1, c2, n, a, dir, max) ∧ n > 1}
Cada tupla del conjunto, ce = (t, n1), identifica el tramo (t = (c1, c2, n, a, dir, max)) y el carril sobre el que estacionan los vehículos (representado por n1). Si n1 = 0, los vehículos estacionan sobre el carril 0, si n1 = 1 lo hacen sobre el carril n-1. En caso de definir 2 tuplas para t, (t, 0) y (t, 1), los vehículos estacionan sobre los 2 bordes de la calle.

3.5. Apéndice C - Resumen del Lenguaje de Especificación de secciones de ciudad

Aquí se presenta un resumen de los elementos del lenguaje de especificación de secciones de ciudad que se define a lo largo de este trabajo, acompañados de una breve descripción de su significado.

Especificación	Descripción
Tramos = { (p1, p2, n, a, dir, max) / p1, p2 ∈ Puntos ∧ n, max ∈ N ∧ a, dir ∈ {0, 1} }	Cada elemento representa un sector de una calle exclusiva para circulación de autos.
Cruces = { (c, maxc) / maxc ∈ N ∧ ∃ t ,t' ∈ Tramos ∧ t = (p1, p2, n, a, dir, max) ∧ t' = (p1', p2', n', a', dir', max') ∧ t ≠ t' ∧ (p1 = c ∨ p2 = c) ∧ (p1' = c ∨ p2' = c) }	Cada elemento representa lugares donde se cruzan 2 ó más tramos de circulación de autos.
CrucesSemáforos = { c / c ∈ Cruces }	Cada elemento representa una esquina con semáforos.
RedDeVías = { Vías / Vías = { (t, ℓ, seq) / t ∈ Tramos ∧ ℓ ∈ N ∧ seq ∈ N } }	Cada elemento representa el trazado de las vías de algún ramal de trenes.
Obras = { (t, ni, ℓ, #n) / t ∈ Tramos ∧ t = (c1, c2, n, a, dir, max) ∧ ni ∈ [0, n-1] ∧ ℓ ∈ N ∧ #n ∈ [1, n+1-ni] ∧ #n ≡ 1 mod 2 }	Cada elemento representa un sector del tramo en obras.
$Baches_T$ = { (t, n1, ℓ) / t ∈ Tramos ∧ t = (c1, c2, n, a, dir, max) ∧ n1∈ [0, n-1] ∧ ℓ ∈ N }	Cada elemento representa un tramo con bache.
$Baches_C$ = { c / c ∈ Cruces }	Cada elemento representa un cruce con bache.
ElementosDeControl = { (t, e, ℓ) / t ∈ Tramos ∧ ℓ ∈ N ∧ e ∈ {elevación transversal, depresión transversal, bocacalles, irregularidad continua, señal de PARE, señal de Escuela} }	Cada elemento representa un tramo con algún elemento de control.
AutosEstacionados = { (t, n1) / t ∈ Tramos ∧ n1 ∈ { 0, 1 } ∧ t = (c1, c2, n, a, dir, max) ∧ n > 1}	Cada elemento representa un tramo con vehículos que estacionan sobre los carriles del borde.
TramosIngreso = { t / t = (p1, p2, n, a, dir, max) ∧ t ∈Tramos ∧ [(dir = 0 ∧ (O v ∈ N : (p2,v) ∈ Cruces)) ∨ (dir = 1 ∧ (O v ∈ N : (p1,v) ∈ Cruces))] }	Cada elemento representa un tramo que actúa como punto de ingreso de vehículos en la simulación.
TramosSalida = { t / t = (p1, p2, n, a, dir, max) ∧ t ∈ Tramos ∧ [(dir = 0 ∧ (O v ∈ N : (p1,v) ∈ Cruces)) ∨ (dir = 1 ∧ (O v ∈ N : (p2,v) ∈ Cruces))] }	Cada elemento representa un tramo que actúa como punto de salida de vehículos de la simulación.
TramosCamiones = { (p1, p2, n, a, dir, max) / p1, p2 ∈ Puntos ∧ n, max ∈ N ∧ a, dir ∈ {0, 1} }	Cada elemento representa un sector de una calle que permite circulación de autos y camiones.
CrucesCamiones = { (c, maxc) / maxc ∈ N ∧ ∃ t ,t' ∈ (TramosCamiones U Tramos) ∧ t = (p1, p2, n, a, dir, max) ∧ t' = (p1', p2', n', a', dir', max') ∧ t ≠ t' ∧ (p1 = c ∨ p2 = c) ∧ (p1' = c ∨ p2' = c) }	Cada elemento representa lugares donde se cruzan 2 ó más tramos de circulación de autos y camiones.
TramosChoques = { t / t ∈ TramosCamiones }	Cada elemento representa un tramo con generación de choques de vehículos.

3.6. Apéndice D - Lenguaje de Especificación para los Modelos

La sintaxis del lenguaje usado para la especificación del comportamiento de los modelos celulares atómicos se define con la siguiente BNF:

RULELIST	= RULE | RULELIST
RULE	= RESULT RESULT { BOOLEXP }
RESULT	= CONSTANT | { REALEXP }
BOOLEXP	= BOOL | { BOOLEXP } | REALRELEXP | not BOOLEXP | BOOLEXP OP_BOOL BOOLEXP
OP_BOOL	= and | or | xor | imp | eqv
REALRELEXP	= REALEXP OP_REL REALEXP | COND_REAL_FUNC(REALEXP)
REALEXP	= IDREF | (REALEXP) | REALEXP OPER REALEXP
IDREF	= CELLREF | CONSTANT | FUNCTION | portvalue(PORTNAME)
CONSTANT	= INT | REAL | CONSTFUNC | ?
FUNCTION	= UNARY_FUNC(REALEXP) | WITHOUT_PARAM_FUNC | BINARY_FUNC(REALEXP, REALEXP) | if(BOOLEXP, REALEXP, REALEXP) | SEND
CELLREF	= (INT, INT)
BOOL	= t | f | ?
OP_REL	= != | = | > | < | >= | <=
OPER	= + | - | * | /
INT	= [SIGN] DIGIT { DIGIT }
REAL	= INT | [SIGN] { DIGIT } . DIGIT { DIGIT }
SIGN	= + | -
DIGIT	= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
PORTNAME	= thisPort | STRING
STRING	= LETTER { LETTER }
LETTER	= a | b | c | ... | z | A | B | C | ... | Z
CONSTFUNC	= pi | e | inf | grav | accel | light | planck | avogadro | faraday | rydberg | euler_gamma | bohr_radius | boltzmann | bohr_magneton | golden | catalan | amu | electron_charge | pem | ideal_gas | stefan_boltzmann | proton_mass | electron_mass | neutron_mass
WITHOUT_PARAM_FUNC	= truecount | falsecount | undefcount | time | random | randomSign
UNARY_FUNC	= abs | acos | acosh | asin | asinh | atan | atanh | cos | sec | sech | exp | cosh | fact | fractional | ln | log | round | cotan | cosec | cosech | sign | sinh | statecount | sqrt | tan | tanh | trunc | truncUpper | poisson | exponential | randInt | chi | asec | acotan | asech | nextPrime | radToDeg | DegToRad | nth_prime | acotanh | CtoF | CtoK | KtoC | KtoF | FtoC | FtoK
BINARY_FUNC	= comb | logn | max | min | power | remainder | root | beta | gamma | lcm | gcd | normal | f | uniform | binomial | rectToPolar_r | rectToPolar_angle | polarToRect_x | hip | polarToRect_y
COND_REAL_FUNC	= even | odd | isInt | isPrime | isUndefined

3.7. Apéndice E – Reglas y Vecindario de los tramos con camiones

Aquí se especifican las vecindades y reglas del comportamiento que le corresponde a cada subespacio de cada tramo con circulación de camiones de acuerdo a la cantidad de carriles que presenta, sin considerar a las celdas borde cuyo comportamiento se encuentra descripto en la sección 2.3.1. Allí también se encuentran definidas las reglas utilizadas en este apéndice.

Calles de carril único (TC1)

El vecindario para las celdas de este espacio es:

N = { (0,-1), (0,0), (0,1), (0,2) };

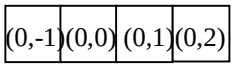


Figura 51 - Vecindario de la celda origen

Y las reglas de movimiento para los vehículos son:

- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás: para esta regla se cambia la macro EsCabeza por EsCabeza_TC1.
- Sale_Auto_Hacia_Adelante
- Sale_Camión_Hacia_Adelante: para esta regla se cambia la macro EsCabeza por EsCabeza_TC1.
- Default

Donde,

- EsCabeza_TC1(i,j) es una macro que se fija si la posición (i,j) es la cabeza o trompa del camión; para eso se chequea que en las posiciones de adelante no haya un vecino que tenga otra parte del mismo camión. Esta macro tiene en cuenta que la celda (i,j) se encuentra en un tramo de único carril. Es decir,

  EsCabeza_TC1(i,j) ≡ (i,j+1) != (i,j) and (i,j+2) != (i,j)

Calles de 2 carriles (TC2)

Las celdas de cada fila de este espacio tienen comportamiento distinto, determinado por la conformación de su vecindario, y serán especificadas por separado.

Para las celdas del carril 0 se define su vecindad como:

N = { (0,-1), (0,-2), (0,-3), (0,-4), (0,-5), (0,0), (0,1), (0,2), (-1,2), (-1,1), (-1,0), (-1,-1), (-1,-2), (-1,-3), (-1,-4), (-1,-5) };

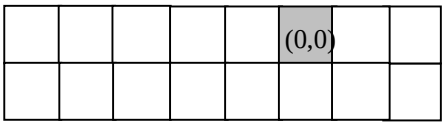


Figura 52 – Vecindario de la celda origen

Y sus reglas de movimiento para los vehículos son:

- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás: para esta regla se cambia la macro EsCabeza por EsCabeza_PrimCarril.
- Llega_Auto_Desde_CarrilDer_TC2
- Llega_Camión_Desde_CarrilDer_TC2
- Sale_Auto_Hacia_Adelante
- Sale_Camión_Hacia_Adelante: para esta regla se cambia la macro EsCabeza por EsCabeza_PrimCarril.
- Sale_Auto_Hacia_CarrilDer_TC2
- Sale_Camión_Hacia_CarrilDer: para esta regla se cambia la macro EsCabeza por EsCabeza_PrimCarril.
- Default

Donde las reglas Llega_Auto_Desde_CarrilDer_TC2, Llega_Camión_Desde_CarrilDer_TC2 y Sale_Auto_Hacia_CarrilDer_TC2 se obtienen a partir de Llega_Auto_Desde_CarrilDer, Llega_Camión_Desde_CarrilDer y Sale_Auto_Hacia_CarrilDer adecuándolas según el vecindario de este carril de TC2, respectivamente. Aquí:

Nuevo estado	Estado Vecindario	Delay	d	Nombre
1	(0,0) = 0 and (-1,-1) = 1 and (-1,0) != 0 and (0,-1) = 0	transport	Conversión_Demora (velocidad)	Llega_Auto_Desde_CarrilDer_TC2
(-1,0)	(0,0) = 0 and HayCamión(-1,0) and (-1,1) != 0 and EsCabeza_UltCarril(-1,0) and CarrilIzLibre(-1,0)	transport	Conversión_Demora (veloc_camión(max))	Llega_Camión_Desde_CarrilDer_TC2
(-1,0)	(0,0) = 0 and HayCamión(-1,0) and (0,1) = (-1,0)	transport	0	
0	(0,0) = 1 and (-1,0) = 0 and (-1,1) = 0 and not(HayCamión(0,1))	transport	Conversión_Demora (velocidad(max))	Sale_Auto_Hacia_CarrilDer_TC2

Y además,

- EsCabeza_PrimCarril(i,j) es una macro que se fija si la posición (i,j) es la cabeza o trompa del camión; para eso se chequea que en las posiciones de adelante no haya un vecino que tenga otra parte del mismo camión. Esta macro tiene en cuenta que la celda (i,j) se encuentra en el primer carril del tramo, por esto no chequea el estado del vecino (i+1,j+1). Es decir,

  EsCabeza_PrimCarril(i,j) ≡ (i,j+1) != (i,j) and (i,j+2) != (i,j) and (i-1,j+1) != (i,j)
- CarrilIzLibre(i,j) es una macro que verifica que en el carril izquierdo de la celda (i,j) haya suficiente lugar para que el camión se pueda desplazar hacia allí. Esto es:

  CarrilIzLibre(i,j) ≡ (i,j-remainder((i,j),10)) = 0 and
  ∀k: 0 ≤ k ≤ remainder((i,j),10) ⇒ (i+1,j-k) = 0

Para las celdas del carril 1 se define su vecindad como:

N = { (0,-1), (0,-2), (0,-3), (0,-4), (0,-5), (0,0), (0,1), (0,2), (1,2), (1,1), (1,0), (1,-1), (1,-2), (1,-3), (1,-4), (1,-5) };

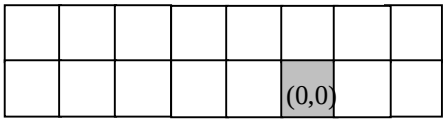


Figura 53 -Vecindario de la celda origen

Y sus reglas de movimiento para los vehículos son:

- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás: para esta regla se cambia la macro EsCabeza por EsCabeza_UltCarril.
- Llega_Auto_Desde_CarrilIz_TC2
- Llega_Camión_Desde_CarrilIz: para esta regla se cambia la macro EsCabeza por EsCabeza_PrimCarril.
- Sale_Auto_Hacia_Adelante
- Sale_Camión_Hacia_Adelante: para esta regla se cambia la macro EsCabeza por EsCabeza_UltCarril.
- Sale_Auto_Hacia_CarrilIz_TC2
- Sale_Camión_Hacia_CarrilIz_TC2
- Default

Donde las reglas Llega_Auto_Desde_CarrilIz_TC2, Sale_Auto_Hacia_CarrilIz_TC2 y Sale_Camión_Hacia_CarrilIz_TC2 se obtienen a partir de Llega_Auto_Desde_CarrilIz, Sale_Auto_Hacia_CarrilIz y Sale_Camión_Hacia_CarrilIz adecuándolas según el vecindario de este carril de TC2, respectivamente. Aquí:

Nuevo estado	Estado Vecindario	Delay	d	Nombre
1	(0,0) = 0 and (1,-1) = 1 and (1,0) != 0 and not(HayCamión(1,0)) and (0,-1) = 0	transport	Conversión_Demora (velocidad(max))	Llega_Auto_Desde_CarrilIz_TC2
0	(0,0) = 1 and (1,0) = 0 and (1,1) = 0	transport	Conversión_Demora (velocidad(max))	Sale_Auto_Hacia_CarrilIz_TC2
0	HayCamión(0,0) and EsCabeza_UltCarril(0,0) and CarrilIzLibre(0,0)	transport	Conversión_Demora (veloc_camión(max))	Sale_Camión_Hacia_CarrilIz_TC2
0	HayCamión(0,0) and (1,1) = (0,0)	transport	0	

Y además,

- EsCabeza_UltCarril(i,j) es una macro que se fija si la posición (i,j) es la cabeza o trompa del camión; para eso se chequea que en las posiciones de adelante no haya un vecino que tenga

otra parte del mismo camión. Esta macro tiene en cuenta que la celda (i,j) se encuentra en el último carril del tramo, por esto no chequea el estado del vecino (i-1,j+1). Es decir,

EsCabeza_UltCarril(i,j) ≡ (i,j+1) != (i,j) and (i,j+2) != (i,j) and (i+1,j+1) != (i,j)

Calles de 3 carriles (TC3)

Las celdas de cada fila de este espacio tienen comportamiento distinto, determinado por la conformación de su vecindario, y serán especificadas por separado.

Para las celdas del carril 0 se define su vecindad como:

N = { (0,-5), (0,-4), (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2), (-1,-5), (-1,-4), (-1,-3), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2), (-2,-5), (-2,-4), (-2,-3), (-2,-2), (-2,-1), (-2,0), (-2,1) }

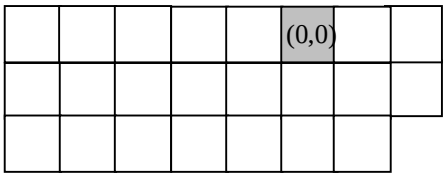


Figura 54 - Vecindario de la celda origen

Y sus reglas de movimiento para los vehículos son:

- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás: para esta regla se cambia la macro EsCabeza por EsCabeza_PrimCarril.
- Llega_Auto_Desde_CarrilDer_TC2
- Llega_Camión_Desde_CarrilDer_TC3
- Sale_Auto_Hacia_Adelante
- Sale_Camión_Hacia_Adelante: para esta regla se cambia la macro EsCabeza por EsCabeza_PrimCarril.
- Sale_Auto_Hacia_CarrilDer
- Sale_Camión_Hacia_CarrilDer: para esta regla se cambia la macro EsCabeza por EsCabeza_PrimCarril.
- Default

Donde la regla Llega_Camión_Desde_CarrilDer_TC3 se obtiene a partir de Llega_Camión_Desde_CarrilDer adecuándola según el vecindario de este carril de TC3. Aquí:

Nuevo estado	Estado Vecindario	Delay	d	Nombre
(-1,0)	(0,0) = 0 and HayCamión(-1,0) and (-1,1) != 0 and EsCabeza(-1,0) and not(CarrilDerLibre(-1,0)) and CarrilIzLibre(-1,0)	transport	Conversión_Demora (veloc_camión)	Llega_Camión_Desde_CarrilDer_TC3
(-1,0)	(0,0) = 0 and HayCamión(-1,0) and (0,1) = (-1,0)	transport	0	

Para las celdas del carril 1 se define su vecindad como:

N = { (1,-5), (1,-4), (1,-3), (1,-2), (1,-1), (1,0), (1,1), (1,2), (0,-5), (0,-4), (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2), (-1,-5), (-1,-4), (-1,-3), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2) }

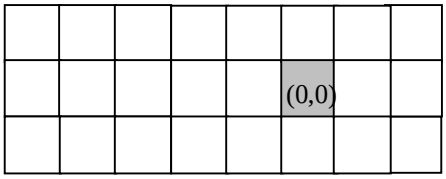


Figura 55 - Vecindario de la celda origen

Y sus reglas de movimiento para los vehículos son:

- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás
- Llega_Auto_Desde_CarrilIz_TC2
- Llega_Camión_Desde_CarrilIz: para esta regla se cambia la macro EsCabeza por EsCabeza_PrimCarril.
- Llega_Auto_Desde_CarrilDer
- Llega_Camión_Desde_CarrilDer_TC3bis
- Sale_Auto_Hacia_Adelante
- Sale_Camión_Hacia_Adelante
- Sale_Auto_Hacia_CarrilIz_TC2
- Sale_Camión_Hacia_CarrilIz_TC3
- Sale_Auto_Hacia_CarrilDer_TC2
- Sale_Camión_Hacia_CarrilDer
- Default

Donde las reglas Llega_Camión_Desde_CarrilDer_TC3bis y Sale_Camión_Hacia_CarrilIz_TC3 se obtienen a partir de Llega_Camión_Desde_CarrilDer y Sale_Camión_Hacia_CarrilIz adecuándolas según el vecindario de este carril de TC3, respectivamente. Aquí:

Nuevo estado	Estado Vecindario	Delay	d	Nombre
(-1,0)	(0,0) = 0 and HayCamión(-1,0) and (-1,1) != 0 and EsCabeza_UltCarril(-1,0) and 2CarrilesIzLibres(-1,0)	transport	Conversión_Demora (veloc_camión(max))	Llega_Camión_Desde_CarrilDer_TC3bis
(-1,0)	(0,0) = 0 and HayCamión(-1,0) and (0,1) = (-1,0)	transport	0	
0	HayCamión(0,0) and EsCabeza(0,0) and CarrilIzLibre(0,0)	transport	Conversión_Demora (veloc_camión(max))	Sale_Camión_Hacia_CarrilIz_TC3
0	HayCamión(0,0) and (1,1) = (0,0)	transport	0	

Para las celdas del carril 2 se define su vecindad como:

N = {(2,-5), (2,-4), (2,-3), (2,-2), (2,-1), (2,0), (2,1), (1,-5), (1,-4), (1,-3), (1,-2), (1,-1), (1,0), (1,1), (1,2), (0,-5), (0,-4), (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2) }

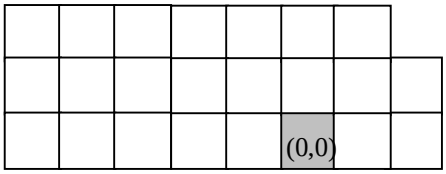


Figura 56 - Vecindario de la celda origen

Y sus reglas de movimiento para los vehículos son:

- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás: para esta regla se cambia la macro EsCabeza por EsCabeza_UltCarril.
- Llega_Auto_Desde_CarrilIz_TC3
- Llega_Camión_Desde_CarrilIz
- Sale_Auto_Hacia_Adelante
- Sale_Camión_Hacia_Adelante: para esta regla se cambia la macro EsCabeza por EsCabeza_UltCarril.
- Sale_Auto_Hacia_CarrilIz
- Sale_Camión_Hacia_CarrilIz: para esta regla se cambia la macro EsCabeza por EsCabeza_UltCarril.
- Default

Donde la regla Llega_Auto_Desde_CarrilIz_TC3 se obtiene a partir de Llega_Auto_Desde_CarrilIz adecuándola según el vecindario de este carril de TC3. Aquí:

Nuevo estado	Estado Vecindario	Delay	d	Nombre
1	(0,0) = 0 and (1,-1) = 1 and (1,0) != 0 and not(HayCamión(1,0)) and (0,-1) = 0 and ((2,-1) != 0 or (2,0) != 0)	transport	Conversión_Demora (velocidad(max))	Llega_Auto_Desde_CarrilIz_TC3

Calles de 4 carriles (TC4)

Las celdas de cada fila de este espacio tienen comportamiento distinto, determinado por la conformación de su vecindario, y serán especificadas por separado.

Para las celdas del carril 0 se define su vecindad y comportamiento en forma análoga al carril 0 de TC3.

Para las celdas del carril 1 se define su vecindad como:

N = { (1,-5), (1,-4), (1,-3), (1,-2), (1,-1), (1,0), (1,1), (1,2), (0,-5), (0,-4), (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2), (-1,-5), (-1,-4), (-1,-3), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2), (-2,-5), (-2,-4), (-2,-3), (-2,-2), (-2,-1), (-2,0), (-2,1) }

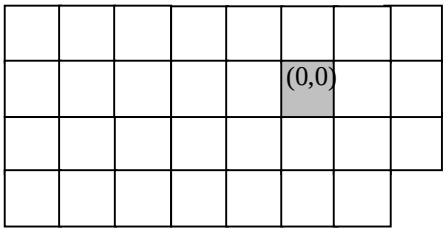


Figura 57 – Vecindario

Y sus reglas de movimiento para los vehículos son:

- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás
- Llega_Auto_Desde_CarrilIz_TC2
- Llega_Camión_Desde_CarrilIz: para esta regla se cambia la macro EsCabeza por EsCabeza_PrimCarril.
- Llega_Auto_Desde_CarrilDer
- Llega_Camión_Desde_CarrilDer
- Sale_Auto_Hacia_Adelante
- Sale_Camión_Hacia_Adelante
- Sale_Auto_Hacia_CarrilIz_TC2
- Sale_Camión_Hacia_CarrilIz_TC3
- Sale_Auto_Hacia_CarrilDer
- Sale_Camión_Hacia_CarrilDer
- Default

Para las celdas del carril 2 se define su vecindad como:

N = { (2,-5), (2,-4), (2,-3), (2,-2), (2,-1), (2,0), (2,1), (1,-5), (1,-4), (1,-3), (1,-2), (1,-1), (1,0), (1,1), (1,2), (0,-5), (0,-4), (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2), (-1,-5), (-1,-4), (-1,-3), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2) }

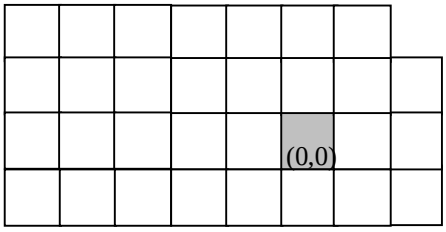


Figura 58 – Vecindario

Y sus reglas de movimiento para los vehículos son:

- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás
- Llega_Auto_Desde_CarrilIz_TC3
- Llega_Camión_Desde_CarrilIz
- Llega_Auto_Desde_CarrilDer
- Llega_Camión_Desde_CarrilDer_TC3bis
- Sale_Auto_Hacia_Adelante

- Sale_Camión_Hacia_Adelante
- Sale_Auto_Hacia_CarrilIz
- Sale_Camión_Hacia_CarrilIz
- Sale_Auto_Hacia_CarrilDer_TC2
- Sale_Camión_Hacia_CarrilDer
- Default

Para las celdas del carril 3 se define su vecindad y comportamiento en forma análoga al carril 2 del tramo de 3 carriles, salvo que se agrega el vecino (3,0) y la regla Llega_Auto_Desde_CarrilIz_TC3 cambia por Llega_Auto_Desde_CarrilIz.

Calles de 5 carriles (TC5)

Las celdas de cada fila de este espacio tienen comportamiento distinto, determinado por la conformación de su vecindario, y serán especificadas por separado.

Para las celdas del carril 0 se define su vecindario y comportamiento en forma análoga al carril 0 de TC4.

Para las celdas del carril 1 se define su vecindad y comportamiento en forma análoga al carril 1 de TC4.

Para las celdas del carril 2 se define su vecindad como:

N = { (2,-5), (2,-4), (2,-3), (2,-2), (2,-1), (2,0), (2,1), (1,-5), (1,-4), (1,-3), (1,-2), (1,-1), (1,0), (1,1), (1,2), (0,-5), (0,-4), (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2), (-1,-5), (-1,-4), (-1,-3), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2), (-2,-5), (-2,-4), (-2,-3), (-2,-2), (-2,-1), (-2,0), (-2,1) }

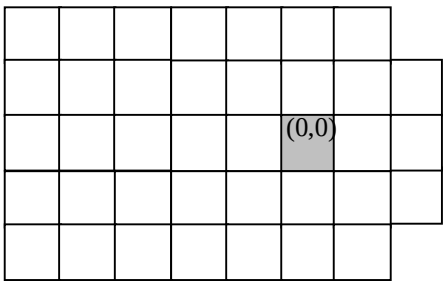


Figura 59 – Vecindario

Y sus reglas de movimiento para los vehículos son:
- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás
- Llega_Auto_Desde_CarrilIz_TC3
- Llega_Camión_Desde_CarrilIz
- Llega_Auto_Desde_CarrilDer
- Llega_Camión_Desde_CarrilDer
- Sale_Auto_Hacia_Adelante
- Sale_Camión_Hacia_Adelante
- Sale_Auto_Hacia_CarrilIz
- Sale_Camión_Hacia_CarrilIz
- Sale_Auto_Hacia_CarrilDer

- Sale_Camión_Hacia_CarrilDer
- Default

Para las celdas del carril 3 se define su vecindad y comportamiento en forma análoga al carril 2 del tramo de 4 carriles, salvo que se agrega el vecino (3,0) y la regla Llega_Auto_Desde_CarrilIz_TC3 cambia por Llega_Auto_Desde_CarrilIz.

Para las celdas del carril 4 se define su vecindad y comportamiento en forma análoga al carril 3 de TC4.

Calles de más de 5 carriles (TC)

Las celdas de cada fila de este espacio tienen comportamiento distinto, determinado por la conformación de su vecindario, y serán especificadas por separado.

Para las celdas del carril 0 se define su vecindad y comportamiento en forma análoga al carril 0 de TC5.

Para las celdas del carril 1 se define su vecindad y comportamiento en forma análoga al carril 1 de TC5.

Para las celdas del carril 2 se define su vecindad y comportamiento en forma análoga al carril 2 de TC5.

Para las celdas de los carriles { i / 3 ≤ i≤ #c-3 } se define su vecindad como:

N = {(3,0), (2,-5), (2,-4), (2,-3), (2,-2), (2,-1), (2,0), (2,1), (1,-5), (1,-4), (1,-3), (1,-2), (1,-1), (1,0), (1,1), (1,2), (0,-5), (0,-4), (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2), (-1,-5), (-1,-4), (-1,-3), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2), (-2,-5), (-2,-4), (-2,-3), (-2,-2), (-2,-1), (-2,0), (-2,1) }

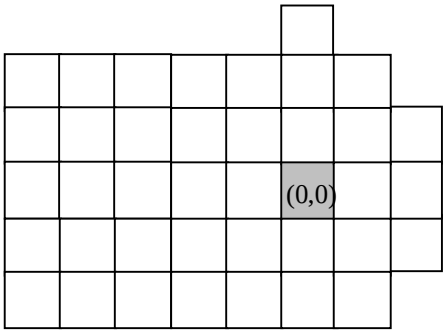


Figura 60 – Vecindario

Y sus reglas de movimiento para los vehículos son:
- Llega_Auto_Desde_Atrás
- Llega_Camión_Desde_Atrás
- Llega_Auto_Desde_CarrilIz
- Llega_Camión_Desde_CarrilIz
- Llega_Auto_Desde_CarrilDer
- Llega_Camión_Desde_CarrilDer

- Sale_Auto_Hacia_Adelante
- Sale_Camión_Hacia_Adelante
- Sale_Auto_Hacia_CarrilIz
- Sale_Camión_Hacia_CarrilIz
- Sale_Auto_Hacia_CarrilDer
- Sale_Camión_Hacia_CarrilDer
- Default

Para las celdas del carril #c-2 se define su vecindad y comportamiento en forma análoga al carril 3 de TC5.

Para las celdas del carril #c-1 se define su vecindad y comportamiento en forma análoga al carril 4 de TC5.

Conclusiones

En el presente trabajo se han analizado varios modelos especificados con el formalismo de Autómatas Celulares, éste ha resultado ser de gran utilidad para describir en forma simple el comportamiento complejo de los sistemas.

Los Autómatas Celulares son aplicables a una gran variedad de problemas, entre ellos se ha profundizado el estudio del control de tráfico. Estos modelos se caracterizan por comenzar especificando los aspectos básicos de la circulación de vehículos, y luego se extienden a través del agregado de reglas que los acercan cada vez más a la realidad. Esta forma de trabajo es favorecida por la naturaleza simple de los Autómatas Celulares que permite el modelado incremental.

Sin embargo, la mayoría de los modelos analizados representan sólo las características elementales del flujo de autos, simplificando la complejidad del problema. Sólo en algunos de ellos se plantean soluciones genéricas que, por ejemplo, representen cualquier cantidad de carriles, puesto que implica un incremento de la complejidad del modelo. Otra simplificación utilizada frecuentemente es el modelado de bordes del AC conectados, que implica un condicionamiento del flujo de vehículos.

Se ha propuesto una taxonomía para facilitar el análisis de los modelos de control de tráfico estudiados y a través de ella se puede establecer qué características del sistema real fueron especificadas. Para su definición se han considerado los aspectos principales que influencian el comportamiento de los vehículos y que deberían ser modelados para obtener resultados más valiosos. Además, la clasificación de los modelos permite descubrir áreas del problema que no han sido representadas.

Se ha definido un lenguaje de alto nivel para la especificación de secciones de ciudad que permite representar una gran variedad de características del tráfico urbano. Se han cubierto aspectos tales como circulación de autos y camiones, presencia de semáforos, choques, baches, etc. Es decir, se ha propuesto una aproximación más completa que la mayoría de los modelos existentes para la simulación de tráfico.

El lenguaje provee un nivel de abstracción que separa los modelos de simulación de la especificación del problema, permitiendo plantear secciones de ciudad de características diversas que se resuelven a través de un mapeo genérico definido para tal fin. Por lo tanto, el usuario del lenguaje no necesita interactuar con los modelos de simulación y por ende no es necesario que conozca los formalismos utilizados para especificarlos.

El lenguaje definido ha sido mapeado sobre los formalismos Cell-DEVS y DEVS. Para ello se han construido modelos que representan cada una de las construcciones del lenguaje, manteniendo los lineamientos básicos que fueron planteados para Autómatas Celulares en trabajos previos. Pero a diferencia de ellos se han agregado más características del tráfico de vehículos y se han realizado la menor cantidad de simplificaciones posibles sobre los modelos planteados. Por ende, se obtuvo un modelo más complejo con mayor cantidad de reglas para describir comportamientos; el cual se espera que genere resultados más precisos.

Por otro lado, el mapeo en modelos DEVS y Cell-DEVS brinda los beneficios de una base formal. Los errores en la simulación se puedan detectar analizando la especificación, sin considerar el sistema de software subyacente. También se reduce el tiempo de resolución del problema, permitiendo analizar comportamientos complejos y brindando nuevas soluciones.

Además, mediante los modelos Cell-DEVS se ha logrado la definición del comportamiento complejo del tráfico de vehículos mediante reglas simples, que se ven favorecidas por las demoras brindadas por el formalismo que se utilizan para representar la velocidad de los autos, modelando este aspecto fuera de las reglas. Por ende, las reglas son de fácil comprensión, modificación y ejecución eficiente; permitiendo la extensión del modelo para incorporar gran cantidad aspectos del problema.

Otro motivo que hace que los formalismos DEVS y Cell-DEVS sean atractivos para sistemas complejos como los de control de tráfico, es que permiten la descripción modular de los fenómenos a modelar. Esta aproximación modular ataca la complejidad del problema, permitiendo resolver en cada modelo algún aspecto del mismo que luego se incorpora en otro de mayor nivel mediante la definición de modelos acoplados. De esta forma es posible concentrarse en un sólo aspecto por vez y definir el modelo que lo represente, para luego establecer su acoplamiento con los demás. Tal es el caso de los cruces y calles que han sido definidos como modelos separados pero que se comunican a través de la interfaz definida para tal fin. También los semáforos y trenes se definieron como modelos separados. Además es posible modificar el comportamiento interno de un modelo, sin que ello afecte a todo el sistema, siempre y cuando las interfaces no cambien.

Los modelos han sido especificados con estado binario, salvo aquellos por los que circulan camiones y se pueden producir choques. Estos últimos se han definido con estado discreto positivo. En los modelos binarios se facilita la validación de las reglas de comportamiento que se puede lograr considerándolas como expresiones booleanas y utilizando operadores binarios únicamente (ejecución eficiente). Se podrían encontrar sin demasiado esfuerzo, inconsistencias en tiempo de compilación, tales como dos reglas distintas que se puedan aplicar y que llevan a diferentes estados. Además, al tratarse de estados binarios la cantidad de memoria necesaria para almacenarlos es pequeña. Pero la gran desventaja frente a los modelos no binarios es que éstos últimos tienen mayor poder expresivo, pues al tener mayor variedad de valores es posible modelar una mayor cantidad de situaciones.

Continuando con las ideas presentadas en este trabajo se pueden plantear nuevas reglas para AC que abarquen aspectos no contemplados en los modelos analizados; para luego poder comparar la precisión de los resultados de estos modelos y de los existentes en función de la complejidad.

Por otro lado, queda pendiente la definición de una interfaz gráfica para usuarios que permita utilizar el lenguaje de especificación para la construcción de secciones de ciudad. Y luego, implementar el mapeo definido entre el lenguaje y los modelos. Además, se puede avanzar para lograr la ejecución paralela de los modelos Cell-DEVS y por ende simulaciones más eficiente.

El modelo de simulación obtenido en este trabajo, resulta ser más completo que los analizados para AC, pero esto también aumenta su complejidad. Por tal motivo, resulta interesante evaluar el aumento de la precisión de los resultados obtenidos en función de la complejidad, considerando aspectos tales como: eficiencia, tiempo de diseño e implementación, etc.

Para profundizar sobre los modelos planteados, se puede extender el comportamiento de los semáforos de forma tal que actúen sincronizadamente a lo largo de una calle o avenida. Algo similar es necesario plantear para la definición de trenes de la sección 1.4.2.2, donde los pasos a nivel deben estar regulados para simular el avance del tren de uno al otro. Por otro lado, el modelado de obras en las calles puede simplificarse mediante el uso de modelos no binarios, siguiendo las ideas planteadas para la representación de los choques.

De acuerdo con lo expuesto, se concluye que el modelo obtenido en el presente trabajo para simulación de tráfico urbano, mejora algunos aspectos de los modelos planteados para AC y que sería de gran interés continuar con este tipo de estudios.

Bibliografía

[BEA97] V. J. BLUE, M. J. EMBRECHTS, J. L. ADLER
"Cellular Automata Modeling of Pedestrian Movements"
0-7803-4053-1/97 IEEE 1997

[BML92] BIHAM, MIDDLETON, LEVINE
Phys. Rev. A46, 6124 (1992)

[CDL97] B. CHOPARD, A. DUPUIS, P. LUTHI
"A Cellular Automata Model for Urban Traffic and its aplications to the city of Genova"
Proceedings of Traffic and Granular Flow
1997.

[CLQ96] B. CHOPARD, P. LUTHI, P. QUELOZ
"Cellular Automata Model of Car Traffic in two-dimensional street networks"
J. Phys. A, vol 29, pp. 2325-2336, 1996.

[CMB93] P. COCKSHOTT, G. MCCASKILL, P. BARRIC
"Use of high speed cellular automata machine to simulate road traffic"
1993.

[CQL95] B. CHOPARD, P. QUELOZ, P. LUTHI
"Traffic Models of a 2D road network"
Proceedings of the 3rd CM users' Meeting
October 1995
Parma.

[EDPWJ89] M. EBLING, DI LORETO, PRESLEY, WIELAND, JEFFERSON
"An ant foraging model implemented on the time warp operating system"
Distributed Simulation 1989. pp. 21.

[GLNW93] A. GREENBERG, B. LUBACHEVSKY, D. NICOL, P. WRIGHT
"Efficient Massively Parallel Simulation of Dynamic Channel Assignment Schemes for Wireless Cellular Communications"
1993.

[MZBG95] Y. MOON, B. P. ZEIGLER, G. BALL, D. P. GUERTIN
"DEVS Representation of Spatially Distributed Systems: Validity, Complexity Reduction"
pp. 288-296.
1995.

[NS92] K. NAGEL, M. SCHRECKENBERG
"A Cellular Automaton Model for Freeway Traffic "
J. Phys, France 2, 2221(1992).

[NSPLDB98] K. NAGEL, P. STRETZ, M. PIECK, S. LECKEY, T. DONNELLY, C. BARRET.
"TRANSIMS traffic flow characteristics"
Transportation Research Board, 77th Annual Meeting
January 11-15, 1998

Washington, D. C.

[NWWS97] K. NAGEL, D. WOLF, P. WAGNER, P. M. SIMON
"Two-lane traffic rules for cellular automata: A systematic approach"
Physical Review E,
1997

[RNSL96] M. RICKERT, K. NAGEL, M. SCHRECKENBERG, A. LAFOUR
"Two line traffic simulation using cellular automata"
Physica A, 231:534, 1996

[SG98] P. M. SIMON , H. A.GUTOWITZ
"A Cellular Automaton Model for Bi-Directional Traffic"
Physical Review E, vol 57:2, pág. 2441-2444.
1998

[SN97] P. M. SIMON, K. NAGEL
"A Simplified Cellular Automata Model for City Traffic"
Reporte Nº 97-4707 LA-UR
LOS ALAMOS National Laboraty.
1997.

[OSH93] B. J. OVEREINDER, P. M. A. SLOOT, L. O. HERTZBERGER
"Time Warp on Transputer Plataform: Pilot Study with Asynchronous Cellular Automata"
1993.

[T96] SHIN-ICHI TADAKY
"Two-dimensional cellular automaton model of traffic flow with open boundaries"
Tech. Rep. Department of Information Science,Saga University.
1996.

[W95] P. WAGNER
"Traffic Simulation using cellular automata: comparison with reality"
Prceedings of Conference Traffic and Granular Flow.
1995

[W96] G. WAINER
"Introducción a la Simulación de Eventos Discretos".
Technical Report 96-005, Departamento de Computación, FCEN/UBA.

[WG98] G. WAINER, N. GIAMBIASI
"Specification, modelling and simulation of timed Cell-DEVS models".
Technical Report 97-007, Departamento de Computación, FCEN/UBA.
Submitted to publication. 1998.

[WG98b] G. WAINER, N. GIAMBIASI
"N-dimensional Cell-DEVS".
Technical Report TR-98-017, Departamento de Computación, FCEN/UBA.

[WNW97] P. WAGNER, K. NAGEL, D. WOLF
"Realistic Multi-line traffic rules for cellular automaton"
Physica A, 234:687, 1997

[Zei76] B. ZEIGLER

"Theory of modeling and simulation".
Wiley, 1976.

Links Asociados

Simulation with Cellular Automata
http://www.tu-bs.de/institute/WiR/weimar/ZAscript/ZAscript.html

Cellular Automata Models
http://cuiwww.unige.ch/~chopard/Traffic/ca-models.html
http://summa.physik.hu-berlin.de/PAPERS/pa3.html

COMPUTATIONAL PHYSICS GROUP
http://rs1.comphys.uni-duisburg.de/

ZPR-Publications: 65C
http://www.zpr.uni-koeln.de/~paper/Keyword_65C.html

The Traffic Simulation Group at the ZPR
http://www.zpr.uni-koeln.de/GroupBachem/VERKEHR.PG/

ZPR-Publications: K. Nagel
http://www.zpr.uni-koeln.de/~paper/Author_knagel.html

Unpublished Papers
http://www.zpr.uni-koeln.de/GroupBachem/VERKEHR.PG/RESEARCH.P/papers/unpublished_papers.html

Howard Gutowitz's Home Page
http://www.santafe.edu/~hag/

Marcus Rickert's Research And Graduate Work
http://www.zpr.uni-koeln.de/~mr/research.html

The TRANSIMS WWW Home Page
http://studguppy.tsasa.lanl.gov/

Simulation Applications (includes TRANSIMS) Papers
http://studguppy.tsasa.lanl.gov/research_team/papers/

CAPÍTULO I - Simulación de Sistemas de Eventos Discretos usando Autómatas Celulares

Una *simulación* es la reproducción del comportamiento dinámico de un sistema real en base a un modelo, con el fin de llegar a conclusiones aplicables al mundo real. Por ende, es el proceso de diseñar un modelo, y conducir experimentos basados en computadoras para describir, explicar y predecir el comportamiento del sistema real [W96].

La utilización de simulaciones se debe a que en muchos casos no se puede experimentar directamente sobre el sistema a estudiar, o se desea evitar costos, peligro, etc. Su empleo permite experimentación controlada, compresión de tiempo (una simulación se realiza en mucho menos tiempo que el sistema real que modela), y análisis de sensibilidad. Otra gran ventaja es que su uso no afecta al sistema real, que puede seguir utilizándose (o no existir). Finalmente, la simulación es una herramienta efectiva de entrenamiento. Algunos problemas que existen al utilizarla son su tiempo de desarrollo, que los resultados pueden tener divergencia con la realidad (precisan validación), y que para reproducir el comportamiento del sistema simulado se precisa una colección extensiva de datos [W96].

Un área de aplicación de la simulación está constituida por los sistemas de control de tráfico de vehículos. La importancia de estos modelos es permitir la evaluación de estrategias, políticas y acciones de transporte, como así también, medir su impacto. Por ejemplo, se pueden testear los efectos de nuevas normas de tránsito y de la introducción de controles en distintas secciones, medir las consecuencias sobre el flujo del vehículo que provoca un choque o parte de una calle con hombres trabajando, controlar la polución ambiental provocada por los automotores, evitar la generación de embotellamientos; etc.

Los objetivos del presente trabajo son analizar diversos modelos que fueron planteados para la simulación del tráfico de vehículos que utilizan el formalismo de Autómatas Celulares y definir una taxonomía con la finalidad de organizar la información e identificar los aspectos importantes que no hayan sido representados.

Un *autómata celular* es un conjunto infinito n-dimensional de celdas ubicadas geométricamente, donde cada punto de la grilla (*celda*) puede tener un estado elegido de un alfabeto finito y evoluciona en el tiempo de acuerdo a un conjunto de *reglas locales* de transición bien definidas. Además todas las celdas contienen el mismo aparato de cálculo del nuevo estado y se conectan entre sí de forma uniforme [W96]. Se define la *vecindad* de una celda, como un conjunto de celdas cercanas, que en general es homogénea para todas y se utiliza como parámetro para obtener su nuevo estado.

Como el espacio de celdas es infinito, para que pueda ser simulado, el modelo debe acotarse a un número finito de celdas en cada paso. La forma más simple de hacerlo es limitando el modelo a un área finita, lo que hace que se pierda homogeneidad: qué se hace con los *bordes*? Para esto suelen usarse dos aproximaciones: o los estados de los bordes se especifican desde afuera (open boundary conditions), o se conectan los extremos entre sí (periodic boundary conditions) [W96].

Dentro del formalismo de autómatas celulares se definen dos variantes cuya diferencia radica primordialmente en cuándo actualizan las celdas sus estados. Por un lado, se definen los *autómatas celulares sincrónicos* que son aquellos donde el estado de las celdas se actualiza simultáneamente en pasos de tiempo discreto. Por el otro lado, están los *autómatas celulares asincrónicos* que son aquellos donde las transiciones de estado de cada celda son independientes

del resto. Para éstos es necesario definir una función que establece cuándo una celda debe actualizar su estado, la misma recibe como parámetros la celda en cuestión, sus vecinos y el tiempo actual.

En el presente trabajo se analiza la utilización de autómatas celulares para modelar el comportamiento del tráfico. Además, se analizan otras aplicaciones de autómatas celulares con menor nivel de profundización. Tal es el caso del comportamiento simplificado de hormigas que buscan comida, la asignación de canal en un sistema de telefonía celular, el comportamiento simplificado de tiburones y peces (Wa-Tor), el flujo de agua a través de una cuenca y el movimiento de peatones. Para todos los modelos considerados, se ha hecho hincapié en las reglas que rigen el comportamiento del sistema, sin considerar aspectos de implementación de las simulaciones. También se ha realizado una formalización de los modelos, de acuerdo a la definición de autómatas celulares incluida en la sección 4.

4. Autómatas Celulares

En esta sección se incluye una definición formal para los paradigmas de autómatas celulares presentada en [WG98]. Se comienza analizando autómatas celulares conceptuales, y luego se definen autómatas celulares ejecutables (sincrónicos y asincrónicos).

Un autómata celular conceptual puede definirse como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

donde

S es el alfabeto utilizado para representar el estado de cada celda;

n es la dimensión del espacio de celdas;

C es la definición del conjunto de estados para el espacio de celdas;

η es el tamaño de la vecindad;

N es el conjunto de vecindad;

T es la función de transición global;

τ es la función de cómputo local; y

$c.\boldsymbol{Z}_0^+$ es la base de tiempo (discreta) del autómata celular.

Analicemos con detalle la definición de cada uno de estos conjuntos:

$\mathbf{S} \subseteq \boldsymbol{Z} \wedge \#S < \infty$;

$\mathbf{n} \in \boldsymbol{N}$;

Notación 5

De aquí en más, se llamará $C_c \in S$ al estado de la celda c, siendo $c \in \boldsymbol{Z}^n$, $c = (i_1,...,i_n)$ la posición de la celda c en el espacio n-dimensional. Aquí, $\forall\ k \in [1,n]$, $i_k \in \boldsymbol{Z}$ es la posición de la celda en la k-ésima dimensión.

Para el caso de autómatas celulares conceptuales, $\forall\ k \in [1,n]$, $i_k \in [-\infty,\infty]$.

$\mathbf{C} = \{ C_c / c \in \boldsymbol{Z}^n \wedge C_c \in S \};$

Para el caso de autómatas celulares bidimensionales [WG98b],

$C = \{ C_{ij} / i,j \in \boldsymbol{Z} \wedge C_{ij} \in S\};$

$\eta \in \boldsymbol{N};$

$\mathbf{N} = \{ (v_{k1},...,v_{kn}) / \forall (k \in \boldsymbol{N}, k \in [1, \eta]) \wedge (i \in \boldsymbol{N}, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \};$

$\mathbf{T}$: $C \times c.\boldsymbol{Z}_0^+ \rightarrow C;$

τ: $N_c \times c.\boldsymbol{Z}_0^+ \rightarrow C_c\ \forall\ c \in \boldsymbol{Z}^n$. En este caso, $C_c[t+c] = \tau(C_{c+v1}[t],...,C_{c+v\eta}[t])$, donde $t \in c.\boldsymbol{Z}_0^+ \wedge \forall (k \in \boldsymbol{N}, k \in [1, \eta])$, $vk \in N \wedge c+vk = (i_1+v_{k1}, ..., i_n+v_{kn})$.

Para autómatas celulares de dos dimensiones, la función de cálculo local [WG98b] puede definirse como:

τ: $S^\eta \times c.\boldsymbol{Z}_0^+ \rightarrow S$. En este caso, $C_{ij}[t+c] = \tau(C_{i0,j0}[t],..., C_{i\eta, j\eta}[t])$,
con $(ik = i+v_{ki} \wedge jk = j+v_{ki})\ \forall (i, j \in \boldsymbol{Z}) \wedge ((v_{ki}, v_{kj}) \in N, k \in \boldsymbol{N}, k \in [1, \eta])$.

$c.\boldsymbol{Z}_0^+ = \{ 0, c, 2c, 3c, ...\} = \{ i / i \in \boldsymbol{N}, i = c.j \wedge j \in \boldsymbol{N} \}$

Como podemos ver en la definición, el modelo está compuesto de un espacio **n**-dimensional (**C**) de celdas. El espacio de estados progresa en pasos de tiempo discreto, dado que la base de tiempo discreta está definida por $c.\boldsymbol{Z}_0^+$ (un conjunto de valores enteros separados entre sí por una constante de tiempo).

En este caso se incluyen dos definiciones: una considera espacios de n dimensiones en los cuales los vecinos son homogéneos (pero pueden pertenecer a cualquier parte del espacio de celdas). Por otro lado, se incluyen definiciones para autómatas celulares de dos dimensiones con vecindades adyacentes a la celda origen.

El estado de cada celda en el espacio puede tomar un valor entre los de un alfabeto finito (**S**). El dominio del conjunto S puede ser cualquier conjunto finito discreto. De aquí en más se considerará esta definición presentada para el conjunto S, sin pérdida de generalidad en la definición del alfabeto de la celda. Esto se debe a que, a pesar de que puede usarse cualquier conjunto finito de símbolos, se puede hacer un mapeo de estos en el S definido en esta sección.

La vecindad está definida como una lista de η vecinos de n dimensiones. En la definición se usa un índice (k) que permite identificar el número de vecino, y un segundo índice (i) que indica la

dimensión de una de las posiciones del vecino. En este caso, los vecinos son una n-upla de posiciones relativas a la celda origen.

Como vemos, las vecindades consideradas implican autómatas celulares homogéneos, aunque las definiciones pueden extenderse fácilmente para autómatas inhomogéneos. En este caso, N se debe definir como un arreglo de listas de vecindades, especificado formalmente como:

$$N = \{N_c / c \in \boldsymbol{Z}^n\}$$

con

$$N_c = \{ (v_{k1},...,v_{kn})_c / \forall (k \in \boldsymbol{N}, k \in [1, \eta_c]) \wedge (i \in \boldsymbol{N}, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \};$$

Si suponemos el caso de espacios bidimensionales,

$$N = \{N_{kl} / k, l \in \boldsymbol{Z}\}$$

donde

$$N_{kl} = \{(i_p,j_p) / \forall p \in \boldsymbol{N}, p \in [1,\eta_{kl}], i_p, j_p \in \boldsymbol{Z} \wedge \eta_{kl} \in \boldsymbol{N}\}$$

En este caso, cada celda tendrá una lista de vecinos de η_c (η_{kl}) elementos, compuesta por tuplas de índices relativos a la celda de origen.

La evolución del espacio de estados del autómata está definida por la ejecución de una función de transición global (**T**) que cambia el estado de todo el espacio de celdas. El comportamiento de esta función de transición depende del resultado de la ejecución de funciones de cálculo locales (**τ**) que corren en el entorno de la vecindad de una celda (**N**). Conceptualmente, el cálculo de estas funciones locales se hace de forma sincrónica y paralela en cada celda del espacio. En el caso de autómatas celulares bidimensionales, $C_{ij}[t]$ es el estado de la celda (i,j) en el tiempo simulado t, y τ denota la función de cálculo local para el autómata. Los parámetros (i0,j0),..., (iη,jη) definen la vecindad de la celda.

En el caso anterior, los índices del espacio de celdas podían incluir infinitas celdas. Como fuera comentado previamente, estamos interesados en modelos ejecutables. Para lo cual un autómata celular ejecutable sincrónico se define formalmente como:

$$CA = < S, n, \{t_1,...,t_n\}, C, \eta, N, B, T, \tau, c.\boldsymbol{Z}_0^+ >$$

donde

S es el alfabeto utilizado para representar el estado de cada celda;

n es la dimensión del espacio de celdas;

{**t_1,...,t_n**} es la cantidad de celdas en cada una de las dimensiones;

C es la definición del conjunto de estados para el espacio de celdas;

η es el tamaño de la vecindad;

N es el conjunto de vecindad;

T es la función de transición global;

B es el conjunto de celdas de borde;

$\boldsymbol{\tau}$ es la función de cómputo local; y

$c.\boldsymbol{Z}_0^+$ es la base de tiempo.

En este caso,

$\mathbf{S} \subseteq \boldsymbol{Z} \wedge \#S < \infty$;

$\mathbf{n} \in \boldsymbol{N}$; $n < \infty$;

$\{\mathbf{t_1,...,t_n}\} \in \boldsymbol{N}$, con $t_k < \infty \; \forall \; k \in [1,n]$;

$\mathbf{C} = \{ C_c / c \in \mathbf{I} \wedge C_c \in S \}$, con

$$\mathbf{I} = \{ (i_1,...,i_n) / i_k \in \boldsymbol{N} \wedge i_k \in [1, t_k] \; \forall \; k \in [1, n] \} \; (1)$$

Para el caso de autómatas celulares bidimensionales [WG98b], se obtiene
$C = \{ C_{ij} / i,j \in \boldsymbol{N} \wedge i \in [1,f], j \in [1,c]\}$, donde $t_1 = f$ es la cantidad de filas y $t_2 = c$ es la cantidad de columnas;

$\boldsymbol{\eta} \in \boldsymbol{N}$ es el tamaño de la vecindad;

$\mathbf{N} = \{ (v_{k1},...,v_{kn}) / \forall (k \in \boldsymbol{N}, k \in [1, \eta]) \wedge (i \in \boldsymbol{N}, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \wedge t_i - | v_{ki} | \geq 0 \}$;

Para el caso de autómatas celulares bidimensionales con vecindades adyacentes a la celda de origen [WG98b], $\mathbf{N} = \{ (i_p,j_p) / i_p, j_p \in \boldsymbol{Z} \wedge i_p, j_p \in [-1, 1] \; \forall \; p \in \boldsymbol{N}, p \in [1,\eta] \}$.

$\mathbf{B} = \{\varnothing\}$ si el espacio de celdas es toroidal; o
$\mathbf{B} = \{C_b / C_b \in C$ con $b \in \mathbf{I} \}$, con **I** definido como en (1). En este caso, el conjunto B está sujeto a la restricción que $\tau_b \neq \tau_c = \tau \; \forall (C_c \notin B) \wedge (C_b \in B)$.

En general, si $B \neq \{\varnothing\}$, B suele definirse como:

$B = \{C_b / C_b \in C$ con $b \in L\}$, siendo $L = \{ (i_1,...,i_n) / i_j = 0 \vee i_j = t_j \; \forall \; j \in [1, n]\}$, y sujeto a que $\tau_b \neq \tau_c = \tau \; \forall \; c \notin L$.

Para el caso de espacios bidimensionales [WG98b] se obtiene:

- $B = \{\varnothing\}$ si el espacio de celdas es toroidal; o
- $B = \{C_{ij} / C_{ij} \in C \wedge (i = 1 \vee i = f) \wedge (j = 1 \vee j = c) \}$ sino.

$\mathbf{T}$: $C \times c.\boldsymbol{Z}_0^+ \rightarrow C$;

$\boldsymbol{\tau: N_c \times c.Z_0^+ \rightarrow C_c}$**; donde** $\mathbf{C_c[t+c] = \tau(C_{c+v1}[t],...,C_{c+v\eta}[t])}$**, (2)**

con $t \in c.\boldsymbol{Z}_0^+ \wedge \forall (k \in \boldsymbol{N}, k \in [1, \eta]), vk \in N \wedge c+vk = (i_1+v_{k1} \bmod(t_1), \; ..., \; i_n+v_{kn} \bmod(t_n))$ en el caso que $B = \{\varnothing\}$, y

$C_b[t+c] = \tau_b(C_b[t])$, $\forall\ b \in B$, con $t \in c.\boldsymbol{Z}_0^+$. En este caso, $\tau_b \neq \tau_c = \tau\ \forall\ c \notin B$, y para estos últimos, C_c se calcula como en (2).

Como puede verse, la definición de los autómatas celulares ejecutables difiere en algunos aspectos de la de los autómatas celulares conceptuales. Por un lado, se acota el tamaño del espacio de celdas en cada una de las dimensiones ($\mathbf{t_1,...,t_n}$), que en este caso también es finita. Por ende, los subíndices de las celdas también se acotan (números naturales finitos).

Otra restricción de los autómatas ejecutables es la pérdida de homogeneidad del espacio de celdas. Ello implica la inclusión de un conjunto de celdas de borde (**B**), que puede ser vacío (para los autómatas celulares toroidales), o un conjunto de celdas prefijado. Estas celdas tienen distinto comportamiento que el resto, por ende, usan distintas funciones de transición local, que se calculan de forma diferente en los bordes y en el resto del autómata.

Como en el caso de los autómatas celulares conceptuales, la semántica de la función de transición T implica la ejecución simultánea y paralela de las funciones de transición local en todo el autómata celular. Esta puede plantearse formalmente como:

$$C_c \in C\ \ \forall\ c = \{\ (i_1,...,\ i_n) / i_j \in [1,\ t_j\ ]\ \forall\ j \in [1,n]\ \},\qquad t \in c.\boldsymbol{Z}_0^+$$

$$C[t+c] = T(C[t]),\qquad \text{con } C_c[t+c] = \tau(C_c[t])\ \forall\ c = \{\ (i_1,...,\ i_n) / i_j \in [1,\ t_j\ ]\ \forall\ j \in [1,n]\ \}$$

Finalmente, para el caso de autómatas celulares asincrónicos, se puede considerar la siguiente definición formal:

$$ACA = < S,\ n,\ \{t_1,...,t_n\},\ C,\ \eta,\ N,\ B,\ Nevs,\ T,\ \tau,\ \boldsymbol{R}_0^+ >$$

donde

S es el alfabeto utilizado para representar el estado de cada celda;

n es la dimensión del espacio de celdas;

$\{\mathbf{t_1,...,t_n}\}$ es la cantidad de celdas en cada una de las dimensiones;

C es la definición del conjunto de estados para el espacio de celdas;

$\boldsymbol{\eta}$ es el tamaño de la vecindad;

N es el conjunto de vecindad;

B es el conjunto de celdas de borde;

Nevs es una lista de próximos eventos;

T es la función de transición global;

$\boldsymbol{\tau}$ es la función de cómputo local; y

$\boldsymbol{R}_0^+$ es la base de tiempo (continua) del autómata celular.

En este caso,

$\mathbf{S} \subseteq \boldsymbol{Z} \wedge \#S < \infty$;

$\mathbf{n} \in \boldsymbol{N}$; $n < \infty$;

$\{\mathbf{t_1},...,\mathbf{t_n}\} \in \boldsymbol{N}$, siendo $t_k < \infty \; \forall \; k \in [1,n]$;

$\mathbf{C} = \{ C_c / c \in \mathbf{I} \wedge C_c \in S \}$, con **I** definido como en (1);

$\boldsymbol{\eta} \in \boldsymbol{N}$;

$\mathbf{N} = \{ (v_{k1},...,v_{kn}) / \forall (k \in \boldsymbol{N}, k \in [1, \eta]) \wedge (i \in \boldsymbol{N}, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \wedge t_i - | v_{ki} | \geq 0\}$;

B = {∅} si el espacio de celdas es toroidal; o
B = $\{C_b / C_b \in C$ con $b \in \mathbf{I} \}$, con **I** definido como en (1). En este caso, el conjunto B está sujeto a la restricción que $\tau_b \neq \tau_c = \tau \; \forall \; (C_c \notin B) \wedge (C_b \in B)$.

Nevs = { (c, t) / $c \in \mathbf{I} \wedge t \in \boldsymbol{R}_0^+$}, donde c es la posición de una celda en el espacio, **I** está definido como en **I** = { (i1,...,in) / $ik \in \boldsymbol{N} \wedge ik \in [1, tk] \; \forall \; k \in [1, n]$ } (1), y t es la hora del evento correspondiente.

T: C x $\boldsymbol{R}_0^+ \rightarrow$ C;

τ: $\mathbf{N_c}$ x $\boldsymbol{R}_0^+ \rightarrow \mathbf{C_c}$; donde $\mathbf{C_c[t_p] = \tau(C_{c+v1}[t],...,C_{c+v\eta}[t])}$, (3)

con $t \in \boldsymbol{R}_0^+ \wedge \forall (k \in \boldsymbol{N}, k \in [1, \eta])$, $vk \in N \wedge c+vk = (i_1+v_{k1} \bmod(t_1), \; ..., \; i_n+v_{kn} \bmod(t_n))$ en el caso que B = {∅}, y

$C_b[t_p] = \tau_b(C_b[t]), \; \forall \; b \in B$, con $t \in \boldsymbol{R}_0^+$. En este caso, $\tau_b \neq \tau_c = \tau \; \forall \; c \notin B$, y para estos últimos, C_c se calcula como en (3). Aquí, $t_p = \min\{t_i\}_{i=1}^{n}$, con $i, p \in \boldsymbol{N} / \; t_i \in \boldsymbol{R}_0^+$. En este caso, $b = c_p \vee c = c_p$, con $(t_i, c_i) \in$ Nevs con la restricción que $\tau_b \neq \tau_c = \tau \; \forall \; c \notin \mathbf{I}$ (1), y para estos últimos, C_c se calcula como en (3).

Como puede verse, la mayoría de los conjuntos y funciones son similares que para el caso sincrónico. Los cambios se deben a que aquí, la base de tiempo es continua (es decir, que las variables de tiempo $t \in \boldsymbol{R}_0^+$). Esta definición asincrónica provoca que el formalismo deba incluir una lista de próximos eventos (**Nevs**), que debe utilizarse para almacenar la información correspondiente a los eventos a ser tratados.

En este caso, la semántica de la función de cálculo global (**T**) es diferente al caso anterior. Aquí, esta función implica la ejecución un grupo de celdas no latentes, llamadas inminentes. La ejecución de esta función se hace simultáneamente en todas las celdas inminentes para un tiempo simulado dado. Esto puede especificarse como:

$C_c \in C \ \forall c \in \mathbf{I}$ (1), $t_p = \min\{t_i\}_{i=1}^{n}$, con $(t_i, c_i) \in$ Nevs $\Rightarrow C_p = \{ (t_p, c_p) / (t_p, c_p) \in$ Nevs $\}$

$$C[t_p] = T(C[t]), \qquad \text{con } C_{cp}[t_p] = \tau(C_{cp}[t]) \ \forall \ c_p \in C_p \ \wedge \ C_c[t_p] = C_c[t] \ \forall \ c \notin C_p$$

Esta definición puede ser extendida para que el conjunto de estados posibles, S, sea un conjunto de conjuntos de números. Es decir, se representa el estado de una celda como un conjunto de números, que podrían ser enteros o reales. Esto es necesario para permitir el modelado de los sistemas con variables inherentemente reales y de aquellos cuyo estado queda mejor definido con más de sola variable entera. Para estos casos se utilizará la notación que se define a continuación.

Notación 6:

En la descripción de los autómatas celulares para los modelos de la siguientes secciones, se notará como $\mathbf{C_k.s_i}$, al valor de la componente s_i del estado de la k-ésima celda; cuando el estado de una celda está compuesto por un conjunto de valores enteros, es decir, $\mathbf{S} = \{ (s_1, s_2, ..., s_n) / s_i \in Z$ para todo $i \in [1,..,n] \}$.

5. Diversas aplicaciones modeladas con Autómatas Celulares

En esta sección se presentan ejemplos de sistemas reales modelados con autómatas celulares. Se realiza la descripción del comportamiento simplificado de hormigas que buscan comida, la asignación de canal en un sistema de telefonía celular, el comportamiento simplificado de tiburones y peces (Wa-Tor), el flujo de agua a través de una cuenca y el movimiento de peatones.

5.1. Modelado del comportamiento de hormigas

En [EDPWJ89] se plantea un modelo para representar algunos aspectos del comportamiento de una colonia de hormigas llamada pagonomyrmex barbatus como benchmark para un entorno de simulación. Se considera que las hormigas pueden realizar tres tipos de acciones:

- Buscar comida: las hormigas persiguen el olor a comida hasta llegar a la fuente.
- Comer.
- Trophallaxis: las hormigas excitan a otras hormigas para que busquen comida (Este comportamiento se modela sólo cuando las hormigas se hallan en el hormiguero).

Para modelar el problema se utiliza un *autómata celular* representado por una grilla de hexágonos (celdas) de igual tamaño que simulan el suelo por donde caminan las hormigas. Cada borde de los hexágonos corresponde a un punto cardinal, como se indica en la siguiente figura:

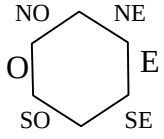


Figura 61 - Celda

Así, cada hexágono constituye su vecindario con 6 *celdas*, una en cada dirección (NO, O, SO, NE, E, SE), que son aquellas con las que comparte un lado.

Cada hexágono puede contener comida, olor a comida y/o una cierta cantidad de hormigas. Todas las *porciones de comida* tienen el mismo aroma, por lo que se suman al llegar aromas de distintas fuentes a la misma celda y se dispersan a igual velocidad a igual cantidad de celdas. Pero cada una puede tener una cantidad de comida distinta. El *aroma a comida* se difunde a hexágonos vecinos en función del tiempo; en una celda se determina usando una función lineal de la distancia y el aroma de la celda que contiene la comida. Además se define un valor crítico por debajo del cual el aroma a comida no es percibido por las hormigas.

Las hormigas se mueven en alguna de las 6 direcciones (NO, O, SO, NE, E, SE) en pasos discretos de un hexágono a otro vecino. Para elegir la dirección se utilizan las siguientes reglas locales:

1) Si una hormiga percibe olor a comida, se moverá hacia donde el aroma sea más fuerte. Si 2 (o más) direcciones contienen los mismos valores de olor a comida, la hormiga avanzará hacia la primera de las direcciones alcanzadas en sentido de las agujas del reloj, comenzando con la dirección NO.
2) Si no hay aroma a comida en los hexágonos vecinos, la hormiga elegirá una de las siguientes cuatro posibilidades (todas con igual probabilidad):
   - Se queda en la misma posición.
   - Se mueve en la misma dirección que la movida anterior.
   - Se mueve a alguna de las dos posiciones hacia adelante y en diagonal respecto de la dirección de su última movida.

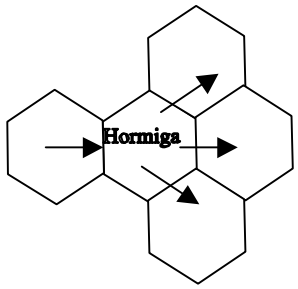


Figura 62 – Movimientos posibles

3) Si una hormiga regresa al hormiguero con comida, provoca que otras hormigas vayan a buscar comida (trophallaxis) y se modifica el número que indica la cantidad de hormigas que fueron en busca de comida.
4) Si una hormiga regresa al hormiguero sin comida, sólo se modifica el número que indica la cantidad de hormigas que fueron en busca de comida.

Un autómata celular que represente este problema puede definirse como:

$$CCA_H = < S_H, n_H, C_H, \eta_H, N_H, T_H, \tau_H, c_H.\boldsymbol{Z}_0^+ >$$

$\mathbf{S}_H = \{ (\#\text{hormigas}, \text{dir}) / \#\text{hormigas} \in \boldsymbol{N}, \text{dir(i)} \in \{1,2,3,4,5,6\}, 1 \leq i \leq \#\text{hormigas} \}$

Donde,

dir(i) indica la dirección de la última movida de la i_ésima hormiga. Si el valor de dir(i) es 1 representa el Noroeste, si es 2 el Suroeste, si es 3 el Este, si es 4 el Oeste, si es 5 el Noreste y si es 6 el Sureste. Para el caso que la celda no tenga ninguna hormiga (#hormigas = 0), no tiene sentido hablar de dir y por lo tanto será representado como indefinido con el símbolo ϕ.

$\mathbf{n}_H = 2$

$\boldsymbol{\eta}_H = 19$

$\mathbf{N}_H$ = { (0,0); (1,-1); (1,1); (0,-1); (0,1); (-1,-1); (-1,1), (2,-1); (2,0); (2,1); (1,-2); (1,2); (0,-2); (0,2), (-1,-2); (-1,2); (-2,-1); (-2,0); (-2,1) }
= { (0,0); (1,-1); (1,1); (0,-1); (0,1); (-1,-1); (-1,1), adyacentes(1,-1); adyacentes(1,1); adyacentes(0,-1); adyacentes(0,1); adyacentes(-1,-1); adyacentes(-1,1) }

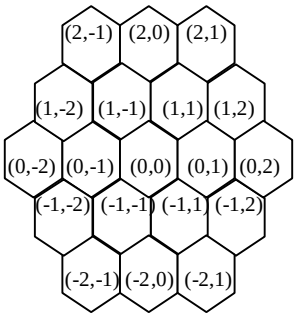


Figura 63 - Vecindario de la celda origen

La función de transición local ($\boldsymbol{\tau}_H$) se define en dos pasos: en el primero, se evalúa la cantidad de hormigas que quedan en la celda origen; en el segundo, se calcula la cantidad de hormigas que ingresan a la celda origen desde las adyacentes. Por último, para saber la cantidad de hormigas de la celda origen, se suman los valores obtenidos en cada uno.

Para establecer una formalización del comportamiento del modelo, se definen las siguientes constantes y funciones:

- La función *adyacentes*, recibe una celda y devuelve a todas las celdas con las que comparte un lado. Por ejemplo, adyacentes(i,j) = { (i+1,j-1); (i+1,j+1); (i,j-1); (i,j+1); (i-1,j-1); (i-1,j+1) }, teniendo en cuenta la siguiente figura con (i,j) = (0,0),

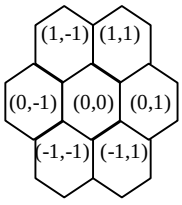


Figura 64 – Celdas adyacentes a (0,0)

- La constante *valor_crítico*, establece el aroma a comida mínimo necesario para ser percibido por una hormiga.
- La función *aroma*, recibe como parámetro a una celda y devuelve el aroma a comida que se percibe en ella. Es decir, aroma(i,j) calcula el aroma de la celda de acuerdo a la cantidad de porciones de comida que se encuentran en la celda y en todas sus adyacentes. Aquí se asume que cada celda conoce la cantidad de comida que posee ella y todas sus vecinas.
- La función *rand*, devuelve un número al azar del conjunto {1, 2, 3, 4}. El 1 indica que la hormiga se queda en la misma celda, el 2 que la hormiga se desplaza en la dirección de la última movida, el 3 que la hormiga se desplaza en diagonal hacia arriba y en la

dirección de la última movida y, por último, el 4 indica que la hormiga se desplaza en diagonal hacia abajo y en la dirección de la última movida

Paso 1: cantidad de hormigas que no se mueven de la celda (0,0)

Nuevo Estado	Estado del vecindario
(0, ϕ)	($C_{(0,0)}$.#hormigas = 0) OR [$C_{(0,0)}$.#hormigas > 0 AND aroma(i,j) > valor_crítico para algún (i,j)∈ adyacentes(0,0)]
(h, v)	$C_{(0,0)}$.#hormigas > 0 AND (∀(i,j)∈ adyacentes(0,0): aroma(i,j) ≤ valor_crítico) AND rand() = 1 para h de las $C_{(0,0)}$.#hormigas de la celda (0,0), cuya dirección de última movida está en v]

La primera regla está formada por 2 subexpresiones, la primera considera el caso en que no hubiera hormigas en la celda y la segunda representa el movimiento de todas las hormigas de la celda origen hacia algún vecino, pues éste contiene suficiente aroma a comida para atraerlas. Por lo tanto, en ambos casos la celda queda sin hormigas.
La segunda regla considera el caso de que no haya ningún vecino con suficiente aroma a comida para atraer a las hormigas de la celda de origen. Por lo tanto, el movimiento de cada hormiga es aleatorio, pero permanecen en la celda aquellas que les toque rand() = 1, caso contrario se desplazan hacia otra celda. Así, se quedarán h hormigas de las totales y la dirección de la última movida se obtiene de $C_{(0,0)}$.dir, construyendo v con aquellas de las hormigas se quedaron.

Paso 2: cantidad de hormigas que avanzan a la celda (0,0), provenientes de todas sus adyacentes.
Cada celda tiene 5 celdas adyacentes desde las que pueden ingresar hormigas. Se denota como (h,k) a alguna de ellas y se definen las reglas para ella. Como pueden ingresar hormigas desde todas las adyacentes es necesario aplicar estas reglas para cada una y se obtiene el resultado final de este paso sumando las cantidades de hormigas que ingresan a la celda origen desde cada adyacente

.

Nuevo Estado	Estado del vecindario
(0, ϕ)	($C_{(h,k)}$.#hormigas = 0) OR [$C_{(h,k)}$.#hormigas > 0 AND (∃(i,j)∈ adyacentes(h,k): (i,j)≠(0,0) AND aroma(0,0) < aroma(i,j) AND aroma(i,j) > valor_crítico)]
($C_{(h,k)}$.#hormigas, v)	$C_{(h,k)}$.#hormigas > 0 AND (∀(i,j)∈ adyacentes(h,k): (i,j)≠(0,0) AND aroma(0,0) > aroma(i,j)) AND aroma(0,0)> valor_crítico AND v es el vector correspondiente a la última movida (depende de qué vecino sea (h,k))
(h2, v)	$C_{(h,k)}$.#hormigas > 0 AND (∀(i,j)∈ adyacentes(h,k): aroma(i,j) ≤ valor_crítico) AND (0,0) está en dirección de la última movida de h1 hormigas de la celda (h,k) AND h2 de las h1 hormigas se desplazan hacia (0,0) de acuerdo a la función rand. AND v es el vector correspondiente a la última movida (depende de qué vecino sea (h,k))
($C_{(h,k)}$.#hormigas, v)	$C_{(h,k)}$.#hormigas > 0 AND aroma(0,0) > valor_crítico AND (∀(i,j)∈ adyacentes(h,k): aroma(0,0) ≥ aroma(i,j)) AND

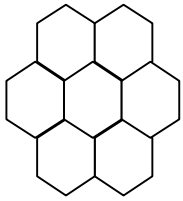
	($\exists$(i,j)$\in$ adyacentes(h,k): (i,j)$\neq$(0,0) AND aroma(0,0) = aroma(i,j)) AND (0,0) es la primera dirección alcanzada en sentido de las agujas del reloj de todas las que tienen el aroma máximo. AND v es el vector correspondiente a la última movida (depende de qué vecino sea (h,k))
(0, ϕ)	Otro caso

La primera regla está formada por 2 subexpresiones, la primera representa que el vecino (h,k) no tiene hormigas, y la segunda representa que todas las hormigas de (h,k) se desplazarán hacia otra adyacente que no es (0,0) pues ésta última tiene menor aroma a comida. Por lo tanto, en ambos casos la cantidad de hormigas que ingresan a la de origen desde la adyacente (h,k) es cero.
La segunda regla representa que todas las hormigas de (h,k) se desplazarán hacia la celda de origen pues ésta tiene mayor aroma a comida que todas las adyacentes de (h,k).
La tercera regla considera el caso de que no haya ningún adyacentes de (h,k) con suficiente aroma a comida para atraer a las hormigas de la celda. Por lo tanto, el movimiento de cada hormiga es aleatorio, y se desplazarán hacia la celda de origen sólo aquellas que la función rand() se lo indique. Así h2 es la cantidad de hormigas que les toca desplazarse en dirección de la última movida y esa dirección corresponda a la celda de origen.
La cuarta regla considera el caso de que tanto la celda de origen como alguna adyacente de (h,k) tengan el aroma máximo a comida para atraer a las hormigas de la celda. Por lo tanto, todas las hormigas se mueven a la primera posición de aroma máximo alcanzada en sentido de las agujas del reloj respecto de (h,k); si la celda de origen cumple esa propiedad entonces todas las hormigas se desplazarán hacia ella.
La quinta regla considera cualquier caso que no cae en los anteriores y en ellos no hay movimiento de hormigas hacia la celda origen.

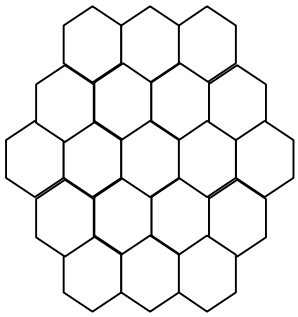
5.2. Asignación dinámica de canales en Telefonía Celular

En [GLNW93] se plantea un modelo para una red de telefonía celular que cubre una cierta región geográfica, donde presta sus servicios. Ese área se divide en *celdas*, con una *estación base* cada una, que reciben solicitudes de canal de usuarios que quieren realizar llamadas. Este pedido se realiza a la estación base más cercana al cliente, la cual decide si lo acepta y qué canal asignarle para la comunicación. Para ello la estación base se comunica con otras estaciones, usando una red cableada independiente. Cuando una llamada es aceptada se le asigna un canal de comunicación con su estación base, quien maneja separadamente la conexión al destinatario. Cuando un llamado es rechazado, se dice que el mismo es bloqueado (es decir, se le niega un canal). Un canal puede ser usado concurrentemente en celdas geográficas distintas siempre y cuando esas celdas estén suficientemente alejadas, para asegurar baja interferencia.

La idea es modelar la asignación dinámica del canal como un autómata cuyas celdas son hexágonos con una estación base en el centro y su vecindad se define como todos los hexágonos dentro de una distancia discreta r de ella. La cantidad de vecinos está dada por la función 3r(r+1). Por ejemplo: si r = 1, las celdas vecinas son las adyacentes a la celda. Si r = 2, las vecinas son las adyacentes a la celda y al primer anillo de celdas vecinas. Estos dos casos son ilustrados en la Figura 65.



Vecindario con
r = 1



Vecindario con r = 2

Figura 65 – Ejemplos de Vecindades

El *fenómeno de interferencia de canales* es un proceso continuo que se modela discretizándolo con el peor caso de propagación de la potencia de la señal en la distancia. Un canal en uso en una celda dada no puede ser asignado a ninguno de sus vecinos, y r se trata como un parámetro fijo relacionado con la potencia y atenuación de la señal.

Para modelar la interferencia y la asignación dinámica de los canales, se usan reglas locales a la celda, pues involucra su estado y el de sus vecinos (y no necesita conocer el estado actual del sistema completo). Se utiliza la técnica de asignación de canales de Markov, que establece que el criterio de aceptación de un llamado depende sólo de la cantidad de llamadas en progreso en cada celda.

Sea M el número total de canales (indexados de 1 a M). Cada celda puede usar todos los canales. El estado de la celda i representa su cantidad de llamadas en progreso, y se denota como *state(i)*. Cada celda i tiene su vector $\nu_i(k)$ de M posiciones, que representa el orden en que los canales son asignados allí. La asignación de canales utiliza una disciplina de pila, tal que si state(i) > 0 entonces los canales en uso en la celda i son $\nu_i(1)$, ..., $\nu_i(\text{state(i)})$; donde $\nu_i(k)$ representa el canal que se adquiere cuando el estado es k. Es decir, si la celda i tiene estado state(i), cuando recibe un pedido de canal, entonces intentará asignar el canal $\nu_i(\text{state(i)} + 1)$ para esa comunicación. Si ese canal está en uso en celdas vecinas entonces el llamado será bloqueado. Las reglas de asignación de canales, teniendo en cuenta las restricciones mencionadas, establecen que si state(i) < M en el momento en que llega una llamada a la celda i y n es el índice del canal correspondiente a $\nu_i(\text{state(i)} + 1)$, entonces la llamada es aceptada en el canal n si éste no está en uso en ningún vecino de i. En caso de aceptarla se incrementa state(i) en 1. Por otro lado, si state(i) < M y el canal $n = \nu_i(\text{state(i)}+1)$ está en uso en alguna celda vecina, o state(i) = M; entonces la llamada es bloqueada (se pierde). También se tiene en cuenta la liberación de los canales asignados al finalizar las llamadas. Si la celda i terminó la llamada del canal $\nu_i(m)$ y m < n (donde, n = state(i)), entonces se introduce momentáneamente un agujero en la pila de canales asignados, $\nu_i(1),...,\nu_i(n),$. Para evitar que esto se mantenga así la última llamada recibida pasa a ocupar el canal $\nu_i(m)$ (el canal liberado por la que salió) y state(i) es decrementado en 1.

Para establecer una formalización del modelo, hay que tener en cuenta que cada celda mantiene cierta información sobre sí misma y sus celdas vecinas que permanece constante durante toda la simulación. Esto se refiere a los vectores de asignación de canales ν, uno por cada vecina y uno propio. Cada ν tiene M posiciones y cada posición del mismo contiene un número de canal, que es un entero del intervalo [1,M], donde M es la cantidad total de canales. ν indica los canales en uso en la celda, que son $\nu(1),..., \nu(\text{state})$; donde state es el estado actual de la celda.

El autómata celular para este problema puede definirse como:

$$CCA_{TC} = < S_{TC}, n_{TC}, C_{TC}, \eta_{TC}, N_{TC}, T_{TC}, \tau_{TC}, c_{TC}.\boldsymbol{Z}_0^+ >$$

$\mathbf{S}_{TC}$ = { n / n es un número entero del intervalo [1,M] }
El estado de la celda indica la cantidad de llamadas en curso en la celda.

$\mathbf{n}_{TC} = 2$

$\boldsymbol{\eta}_{TC} = 7$, asumiendo que r = 1, es decir, el vecindario sólo lo conforman las celdas adyacentes.

$\mathbf{N}_{TC}$ = { (0,0); (1,-1); (1,1); (0,-1); (0,1); (-1,-1); (-1,1) }

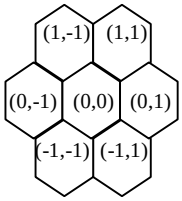


Figura 66 - Vecindario de la celda origen

La función $\boldsymbol{\tau}_{TC}$ se define como:

Nuevo Estado	Estado del vecindario
$C_{(0,0)}$	[$C_{(0,0)}$ = M AND NOT(TerminaLlamada)] OR ($C_{(0,0)}$ < M AND LlegaLlamada AND $\nu(C_{(0,0)}+1)$ está en uso en una celda vecina) OR [NOT (LlegaLlamada) AND NOT(TerminaLlamada)]
$C_{(0,0)} + 1$	LlegaLlamada AND $C_{(0,0)}$ < M AND $\nu(C_{(0,0)}+1)$ no está en uso en ninguna celda vecina
$C_{(0,0)} - 1$	TerminaLlamada

Donde, LlegaLlamada es una función aleatoria que indica la presencia de una llamada en la celda que desea que se le asigne un canal. TerminaLlamada es verdadero cuando se libera un canal de comunicación.
Para verificar que $\nu(C_{(0,0)}+1)$ está o no en uso en alguna celda vecina, se deben revisar los vectores ν de cada vecino (i,j), hasta la posición $C_{(i,j)}$ del mismo. Si el canal buscado no es encontrado, entonces el vecino (i,j) no lo está utilizando. Esta búsqueda es factible, puesto que cada celda conoce los vectores ν de todos sus vecinos y sus correspondientes estados, $C_{(i,j)}$.

5.3. Wa-Tor

El problema de Wa-Tor [OSH93] consiste de un conjunto simple de reglas que describen el comportamiento de tiburones y peces en una porción de océano de forma rectangular. Los peces y tiburones pueden moverse, reproducirse o dormir por algún tiempo. Los tiburones además, se alimentan de los peces, y si durante un cierto tiempo no comen, se mueren por inanición.

El problema Wa-Tor es modelado como un autómata celular de 2 dimensiones, con bordes conectados entre sí. Para representar la reproducción y la muerte por inanición, se utilizan los parámetros, *fbreed* y *sbreed* (edades en que los peces y los tiburones se reproducen, respectivamente); y *starve* (tiempo que un tiburón puede sobrevivir sin comer). El parámetro *tiempo de activar* indica la cantidad de tiempo que un pez o tiburón se queda en una celda luego de haber ejecutado una transición.

Tanto un pez como un tiburón pueden desplazarse a una celda adyacente en dirección norte, este, oeste o sur; en cada paso discreto de tiempo. Las reglas para el movimiento de los peces son simples. Cada pez selecciona una posición al azar entre sus celdas adyacentes vacías. Si no hay posiciones vacías entonces no se mueve. En ambos casos el parámetro *tiempo de activar* toma un nuevo valor, y el pez se queda en la celda durante ese tiempo. Luego que pasó el tiempo correspondiente a *tiempo de activar*, la edad del pez es incrementada con la cantidad de tiempo que el pez permaneció en la posición, y es activado. Si la edad que alcanza un pez es *fbreed*, en el momento que le toca activarse, y además decide moverse, un nuevo pez con edad 0 es colocado en su antigua posición.

Las reglas para el movimiento de los tiburones son similares a las reglas de los peces, excepto que la caza de peces tiene prioridad sobre el mero movimiento. Esto significa que, primero buscan peces en sus posiciones vecinas; eligen alguna de ellas al azar, comen el pez y reinicializan el parámetro que cuenta el tiempo desde la última comida a cero. Si un tiburón nada por una cantidad *starve* de unidades de tiempo sin comer ningún pez, se muere. Luego, cuando no hay peces en las posiciones adyacentes, se mueven de la misma manera que los peces. El *tiempo de activar* toma un nuevo valor, y el tiburón será activado de nuevo luego que pase el tiempo correspondiente, y la edad es incrementada acordemente. Al igual que los peces, los tiburones se reproducen si su edad ha alcanzado *sbreed* en el momento de activación.

El autómata celular para este problema puede definirse como:

$$CCA_{WT} = < S_{WT}, n_{WT}, C_{WT}, \eta_{WT}, N_{WT}, T_{WT}, \tau_{WT}, c_{WT}.\boldsymbol{Z}_0^+ >$$

$\mathbf{S}_{WT}$ = { (clase, edad, última_comida) / clase $\in$ {0,1,2}, edad $\in$ *N*, última_comida $\in$ *N*}
Donde,
clase = 0, indica que la celda está vacía;
clase = 1, indica que la celda está ocupada por un pez; y
clase = 2, indica que la celda está ocupada por un tiburón.
edad, representa la edad del pez o tiburón, siempre y cuando la clase sea 1 ó 2.
última_comida, representa el tiempo en que un tiburón se comió al último pez, la clase debe ser 2.

$\mathbf{n}_{WT}$ = 2

$\boldsymbol{\eta}_{WT}$ = 13

$\mathbf{N}_{WT}$ = { (0,0); (0,-1); (0,1); (1,0); (-1,0); (2,0); (1,-1); (1,1); (0,-2); (0,2); (-1,-1); (-1,1); (-2,0) }

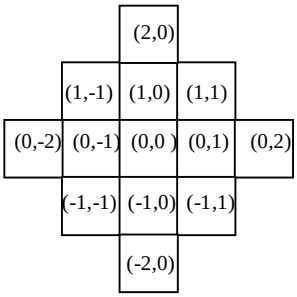


Figura 67 - Vecindario de la celda origen

La función τ_{WT} se define como:

Nuevo Estado	Estado del vecindario
(1, $C_{(0,0)}$.edad+d,0)	$C_{(0,-1)}$.clase > 0 AND $C_{(0,0)}$.clase = 1 AND $C_{(0,1)}$.clase > 0 AND $C_{(1,0)}$.clase > 0 AND $C_{(-1,0)}$.clase > 0 AND d es el tiempo que duerme hasta la próxima activación
(1, e+d,0)	VecinoPezAvanza AND $C_{(0,0)}$.clase = 0 AND d es el tiempo que duerme hasta la próxima activación AND e es la edad del pez que avanza.
(2, $C_{(0,0)}$.edad+d, $C_{(0,0)}$.última_comida)	$C_{(0,0)}$.clase=2 AND TodosVecinosTiburones AND time() - $C_{(0,0)}$.última_comida < starve AND d es el tiempo que duerme hasta la próxima activación
(2, e+d, u)	ExisteVecinoTiburón AND VecinoTiburónAvanzaParaComer AND $C_{(0,0)}$.clase = 1 AND d es el tiempo que duerme hasta la próxima activación AND time() - última_comida < starve para ese vecino tiburón AND e es la edad del tiburón que avanza AND u = time()
(1,0,0)	$C_{(0,0)}$.clase = 1 AND ExisteVecinoVacío AND $C_{(0,0)}$.edad = fbreed
(2,0,0)	$C_{(0,0)}$.clase = 2 AND (ExisteVecinoVacío OR ExisteVecinoPez) AND $C_{(0,0)}$.edad = sbreed AND time() - $C_{(0,0)}$.última_comida < starve
(0,0,0)	[$C_{(0,0)}$.clase = 1 AND ExisteVecinoVacío AND $C_{(0,0)}$.edad ≠ fbreed] OR [$C_{(0,0)}$.clase = 2 AND (ExisteVecinoVacío OR ExisteVecinoPez) AND $C_{(0,0)}$.edad ≠ sbreed AND time() - $C_{(0,0)}$.última_comida < starve] OR [$C_{(0,0)}$.clase = 0 AND TodosVecinosVacíos] OR [$C_{(0,0)}$.clase = 0 AND los vecinos no se mueven hacia ella] OR [$C_{(0,0)}$.clase = 2 AND time() - $C_{(0,0)}$.última_comida ≥ starve]
(2,e+d,u)	$C_{(0,0)}$.clase = 0 AND ExisteVecinoTiburón AND VecinoTiburónAvanzaSinComer AND time() - última_comida < starve para ese vecino AND e es la edad del tiburón que avanza AND u es el tiempo de la última comida del tiburón que avanza AND d es el tiempo que duerme hasta la próxima activación.

Donde,
ExisteVecinoVacío equivale a la siguiente disyunción:
[$C_{(0,-1)}$.clase = 0 OR $C_{(0,1)}$.clase = 0 OR $C_{(1,0)}$.clase = 0 OR $C_{(-1,0)}$.clase = 0]
TodosVecinosVacíos equivale a la siguiente conjunción:
[$C_{(0,-1)}$.clase = 0 AND $C_{(0,1)}$.clase = 0 AND $C_{(1,0)}$.clase = 0 AND $C_{(-1,0)}$.clase = 0]
ExisteVecinoPez equivale a la siguiente disyunción:
[$C_{(0,-1)}$.clase = 1 OR $C_{(0,1)}$.clase = 1 OR $C_{(1,0)}$.clase = 1 OR $C_{(-1,0)}$.clase = 1]
ExisteVecinoTiburón equivale a la siguiente disyunción
[$C_{(0,-1)}$.clase = 2 OR $C_{(0,1)}$.clase = 2 OR $C_{(1,0)}$.clase = 2 OR $C_{(-1,0)}$.clase = 2]
TodosVecinosTiburones equivale a la siguiente conjunción
[$C_{(0,-1)}$.clase = 2 AND $C_{(0,1)}$.clase = 2 AND $C_{(1,0)}$.clase = 2 AND $C_{(-1,0)}$.clase = 2]

“Los vecinos no se mueven hacia ella” equivale a verificar que cada vecino ocupado, no elige desplazarse hacia la celda origen. Esto puede darse porque el vecino es pez y no elige esta celda para desplazarse pues la elección al azar lo lleva a otra, o porque el vecino es tiburón y encuentra un pez en algún otro vecino, o bien, porque no tiene vecinos peces pero al azar no elige la celda origen.
VecinoPezAvanza equivale a que si alguno de los vecinos peces elige desplazarse hacia la celda origen. Esto se debe a que el pez elige a la celda origen en forma aleatoria entre todas sus vecinas vacías.
VecinoTiburónAvanzaParaComer equivale a que si alguno de los vecinos tiburones elige desplazarse hacia la celda origen. Esto se debe a que el tiburón elige a la celda origen en forma aleatoria entre todas sus vecinas con peces.
VecinoTiburónAvanzaSinComer equivale a que si alguno de los vecinos tiburones elige desplazarse hacia la celda origen, considerando que ningún vecino tiene peces, entonces elige entre todas una al azar y resulta ser la celda origen.
time() es una función que devuelve el tiempo actual de la simulación.

5.4. Modelado de una cuenca fluvial

En [MZBG95] se modela el proceso de desagüe de una cuenca fluvial. Ésta se considera formada por varias capas (aire, superficie de agua, superficie de tierra, agua subterránea, capa de rocas), para modelar el comportamiento del agua que circula hacia la desembocadura teniendo en cuenta que parte se infiltra hacia las capas subterráneas.

Todo el terreno que abarca la cuenca se divide en celdas. La lluvia se representa como un flujo de entrada que reciben las celdas del reservorio y se infiltra hacia abajo (capas subterráneas) y lateralmente (hacia celdas vecinas). Las entradas de los vecinos son excesos de agua que representan flujo vertical o lateral. La transición de estado local es un paso de integración de la ecuación diferencial en la cual se basa el modelo hidrológico.

Una celda se representa como tres reservorios conectados verticalmente, el *reservorio de la superficie* que recibe como entradas la lluvia efectiva (r_e) y el flujo (qi(t)) de las celdas vecinas; y que genera como salida el flujo (qo(t)) a celdas vecinas y la infiltración (f(t)) al *reservorio de la subsuperficie*. Y por último, el *reservorio subterráneo* que trabaja de manera similar a la reserva de la superficie a excepción de la infiltración. Estas tres capas constituyen la estructura de cada celda del autómata y los flujos que circulan a través de ellas se muestran en la Figura 68.

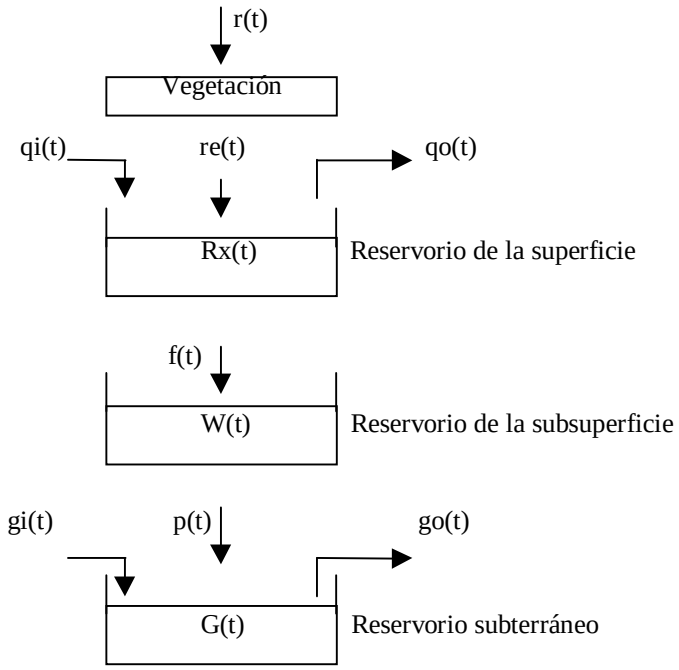


Figura 68 - Estructura de una celda

En la Figura 68 se observa que la entrada de lluvia (*r(t)*) es parcialmente interceptada por una cobertura de vegetación, y el resto de ella, la lluvia efectiva (r_e *(t)*), se infiltra hacia las capas de abajo.

Se define la profundidad de agua sobre una celda (R_x (t)), y el flujo de salida (qo(t)) como:

$$R_x(t) = \int_0^t (r_e(t) + \sum_i qi_i(t) - f(t) - \sum_i qo_i(t))dt \quad (1)$$

$$qo_i(t) = \frac{CS_i(t)R_x(t)}{W_i}$$

Donde,
qi_i (t_i) es el flujo de entrada de la i_ésima celda vecina;
qo_i(t) es el flujo de salida hacia la i_ésima celda vecina;
C es un parámetro que caracteriza la aspereza de la superficie en la ubicación de la celda;
S_i(t) es la pendiente a la i_ésima celda vecina; y
W_i es la distancia a la i_ésima celda vecina.

La pendiente, S_i(t), se calcula como:

$$S_i(t) = \frac{R_x(t) + h - r_{xi}(t) - h_i}{W_i}$$

Donde,
h es la altitud de la celda;
R_{xi} es el exceso de lluvia (o profundidad del agua) de la i_ésima celda vecina; y
h_i es la altitud de la i_ésima celda vecina.

Cada celda puede tener a lo sumo ocho vecinos, una en cada dirección (E, O, N, S) y una en cada diagonal.

Para establecer una formalización del modelo, es necesario considerar que además del estado de la celda y el de sus vecinas, cada celda cuenta con la siguiente información, que es constante durante toda la simulación,

- h que es la altitud de la celda
- $h_{(i,j)}$ que es la altitud del vecino (i,j); (i,j)∈ N
- $W_{(i,j)}$ que es la distancia hasta el vecino (i,j); donde (i,j)∈ N

El autómata celular para este problema puede definirse como:

$$CCA_{CF} = < S_{CF}, n_{CF}, C_{CF}, \eta_{CF}, N_{CF}, T_{CF}, \tau_{CF}, c_{CF}.\mathbf{Z}_0^+ >$$

$\mathbf{S}_{CF} = R_x \in \Re$

Donde, el estado de la celda representa la cantidad de agua acumulada en la misma (R_x).

$\mathbf{n}_{CF} = 2$

$\boldsymbol{\eta}_{CF} = 5$

$\mathbf{N}_{CF}$ = { (1,0); (0,-1); (0,0); (0,1); (−1,0) }

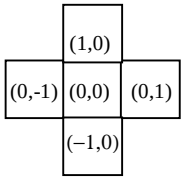


Figura 69 - Vecindario de la celda origen

El nuevo estado de la celda origen ($\boldsymbol{\tau}_{CF}$), se obtiene resolviendo las siguientes ecuaciones:

$$C_{(0,0)}(t) = \int_0^t (r_e(t) + \sum_i qi_i(t) - f(t) - \sum_i qo_i(t))\, dt$$

La primera sumatoria, $\sum_i qi_i(t)$, corresponde a la suma de todos los flujos que ingresan a la celda desde sus vecinos. Cada término, $qi_i(t)$, se obtiene calculando el flujo de salida de la celda vecina hacia la origen. Para cada (i,j) ∈ { (1,0); (0,-1); (0,1); (−1,0) }, se calcula:

$$qo_{(0,0)}(t) = \frac{CS_{(i,j)}(t) C_{(i,j)}(t)}{W_{(i,j)}}$$

$$S_{(i,j)}(t) = \frac{C_{(i,j)}(t) + h_{(i,j)} - C_{(0,0)}(t) - h}{W_{(i,j)}}$$

La segunda sumatoria, $\sum_i qo_i(t)$, corresponde a la suma de todos los flujos que salen de la celda origen hacia sus vecinos. Cada término, $qo_i(t)$, se calcula de forma similar usando las siguientes ecuaciones:

$$qo_{(i,j)}(t) = \frac{CS_{(i,j)}(t) C_{(0,0)}(t)}{W_{(i,j)}}$$

$$S_{(i,j)}(t) = \frac{C_{(0,0)}(t) + h - C_{(i,j)}(t) - h_{(i,j)}}{W_{(i,j)}}$$

5.5. Modelado de movimiento de peatones

En [BEA97] se modela el movimiento de peatones en lugares abiertos con alta densidad de personas. Para ello se utiliza un autómata celular cuyas celdas representan una división del terreno por donde caminan las personas. Cada celda se denota con sus coordenadas en 2 dimensiones. En cada paso de tiempo, T_i, una cantidad de N peatones ingresa al espacio de celdas por alguno de los cuatro bordes y avanzan hacia el lado opuesto. Un peatón busca su camino desde la celda de origen (i_o,j_o) hasta la celda de destino (i_d,j_d) siguiendo reglas locales de comportamiento que evitan colisiones y conflictos entre distintas personas. Como se considera que la densidad de peatones es alta, éstos deben maniobrar alrededor de las celdas ocupadas con alta frecuencia. Para evitar que un peatón quede bloqueado al no poder desplazarse a la celda que le corresponde, se permite en determinadas ocasiones, que una persona fuerce al ocupante de la posición a moverse (lo empuja para que le deje el lugar vacío). En cada paso de tiempo T_i, cada peatón en orden, de acuerdo a su número de entidad, avanza.

Cada peatón que ingresa a la simulación se le asigna un número e_n; una celda de origen (i_o,j_o) y una de destino (i_d,j_d), elegidas al azar. Cada peatón elige una nueva celda para moverse con dirección al destino, en cada paso de tiempo. Considerando a (i_a,j_a) como la posición actual del peatón, si $j_d-j_a=0$ entonces el peatón intenta moverse a la celda que está directamente adelante (i_a+1,j_a). Si está ocupada por otro peatón, se elige con igual probabilidad alguna de las dos diagonales hacia adelante $(i_a+1,j_a\pm1)$, sin salir de la grilla. Si las tres celdas anteriores están ocupadas, entonces hace un *movimiento al costado*.

El movimiento al costado cuando $j_d-j_a=0$ consiste en elegir con igual probabilidad alguno de los dos costados $(i_a,j_a\pm1)$, sin salir de la grilla. Si la celda elegida está ocupada, *empuja* al ocupante. El comportamiento del peatón desplazado es descripto más adelante.

Si $j_d-j_a\neq0$ entonces el peatón elige moverse hacia adelante en diagonal acercándose al destino$(i_a+1,j_a\pm1)$. Si está ocupada por otro peatón, se elige la posición directa hacia adelante (i_a+1,j_a). Si las dos celdas anteriores están ocupadas, entonces hace un movimiento al costado.

El movimiento al costado cuando $j_d-j_a\neq0$ consiste en elegir la celda más cercana a destino, entre sus dos lados $(i_a,j_a\pm1)$, sin salir de la grilla. Si la celda elegida está ocupada, empuja al ocupante.

Por otro lado es necesario definir el comportamiento cuando se produce un empujón, es decir cuando un peatón es echado de su posición. Quien empuja se mueve hacia la posición deseada que tenía otro individuo. Éste se mueve hacia el costado más cercano a destino si $j_d-j_a\neq0$ o hacia cualquier costado si $j_d-j_a=0$. Si la posición elegida está ocupada entonces empuja a su ocupante. Un peatón empujado puede salir de la grilla. El hecho de empujar personas es recursivo y puede resultar en varios empujones antes que una persona encuentre una celda vacía o abandone la grilla.

Un autómata celular para este problema puede definirse como:

$$CCA_{MP} = < S_{MP}, n_{MP}, C_{MP}, \eta_{MP}, N_{MP}, T_{MP}, \tau_{MP}, c_{MP}.\boldsymbol{Z}_0^+ >$$

$\mathbf{S}_{MP}$ = { (dirección, destino) / dirección $\in$ {0,1,2,3,4} $\wedge$ destino $\in$ N }

Donde,

dirección = 0, representa que la celda está vacía;

dirección = 1, representa que en la celda hay un peatón que se dirige hacia arriba;

dirección = 2, representa que en la celda hay un peatón que se dirige hacia abajo;

dirección = 3, representa que en la celda hay un peatón que se dirige hacia la derecha;
dirección = 4, representa que en la celda hay un peatón que se dirige hacia la izquierda; y
destino representa la posición sobre el borde de destino al que debe llegar el peatón.

$\mathbf{n}_{MP} = 2$

$\boldsymbol{\eta}_{MP} = 25$

$\mathbf{N}_{MP}$ = { (0,0); (1,0); (0,-1); (0,1); (–1,0); (1,1); (1,-1); (-1,-1); (-1,1); (2,-2); (2,-1); (2,0); (2,1); (2,2); (1,-2); (1,2); (0,-2); (0,2); (-1,-2); (–1,2); (-2,-2); (-2,-1); (-2,0); (-2,1); (-2,2)}

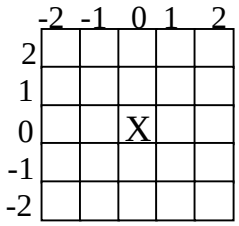


Figura 70 -Vecindario de la celda origen

La función $\boldsymbol{\tau}_{MP}$ se define como:

Nuevo Estado	Estado del vecindario
(0 ,0)	(TodoAdyacenteVacío AND $C_{(0,0)}$.dirección=0) OR $C_{(0,0)}$.dirección>0 OR (NingúnAdyacenteOcupadoAvanza AND $C_{(0,0)}$.dirección=0)
(ocup, dest)	[$C_{(0,0)}$.dirección=0 AND AlgúnAdyacenteOcupadoAvanza AND ese vecino tiene como estado (ocup,dest)] OR [$C_{(0,0)}$.dirección>0 (ocupante echado) AND AlgúnAdyacenteOcupadoAvanzaEchando AND Ese vecino tiene como estado (ocup,dest)]

Donde,

TodoAdyacenteVacío equivale a la siguiente conjunción,
$C_{(1,-1)}$.dirección=0 AND $C_{(1,0)}$.dirección=0 AND $C_{(1,1)}$.dirección=0 AND $C_{(0,-1)}$.dirección=0 AND $C_{(0,1)}$.dirección=0 AND $C_{(-1,-1)}$.dirección=0 AND $C_{(-1,0)}$.dirección=0 AND $C_{(-1,1)}$.dirección=0

AlgúnAdyacenteOcupadoAvanza. Para verificar que esto se cumpla se debe evaluar de las posiciones adyacentes ocupadas cuáles pueden avanzar a la origen, respetando las reglas de movimiento. A continuación se muestra cómo determinar si el ocupante de las posiciones (-1,0), (0,-1)y (1,0) debe va a avanzar a (0,0). Si alguno de los casos es verdadero, entonces el peatón avanzará.

1. $C_{(-1,0)}$.dirección=1 AND $C_{(-1,0)}$.destino-0=0 para ese peatón.
2. $C_{(-1,0)}$.dirección=1 AND $C_{(-1,0)}$.destino-0≠0 para ese peatón AND $C_{(0,1)}$.dirección>0.
3. $C_{(-1,0)}$.dirección=3 AND $C_{(-1,0)}$.destino-0=0 para ese peatón AND rand()=(0,0) AND $C_{(-1,1)}$.dirección>0 AND $C_{(0,1)}$.dirección>0 AND $C_{(-2,1)}$.dirección>0.
4. $C_{(-1,0)}$.dirección=3 AND $C_{(-1,0)}$.destino-0>0 para ese peatón AND $C_{(-1,1)}$.dirección>0 AND $C_{(0,1)}$.dirección>0.

5. $C_{(-1,0)}$.dirección=4 AND $C_{(-1,0)}$.destino-0>0 para ese peatón AND $C_{(0,-1)}$.dirección>0 AND $C_{(-1,-1)}$.dirección>0 AND $C_{(-2,-1)}$.dirección>0.
6. $C_{(-1,0)}$.dirección=4 AND $C_{(-1,0)}$.destino-0≠0 para ese peatón AND rand()=(0,0) AND $C_{(-1,-1)}$.dirección>0 AND $C_{(0,-1)}$.dirección>0

7. $C_{(0,-1)}$.dirección=1 AND $C_{(0,-1)}$.destino-(-1)=0 para ese peatón AND rand()=(0,0) AND $C_{(1,-2)}$.dirección>0 AND $C_{(1,-1)}$.dirección>0 AND $C_{(1,0)}$.dirección>0.
8. $C_{(0,-1)}$.dirección=1 AND $C_{(0,-1)}$.destino-(-1)>0 para ese peatón AND $C_{(1,-1)}$.dirección>0 AND $C_{(1,0)}$.dirección>0.
9. $C_{(0,-1)}$.dirección=2 AND $C_{(0,-1)}$.destino-(-1)=0 para ese peatón AND rand()=(0,0) AND $C_{(-1,-2)}$.dirección>0 AND $C_{(-1,-1)}$.dirección>0 AND $C_{(-1,0)}$.dirección>0.
10. $C_{(0,-1)}$.dirección=2 AND $C_{(0,-1)}$.destino-(-1)>0 para ese peatón AND $C_{(-1,-1)}$.dirección>0 AND $C_{(-1,0)}$.dirección>0.
11. $C_{(0,-1)}$.dirección=3 AND $C_{(0,-1)}$.destino-(-1)=0 para ese peatón
12. $C_{(0,-1)}$.dirección=3 AND $C_{(0,-1)}$.destino-(-1)≠0 para ese peatón AND $C_{(1,0)}$.dirección>0

13. $C_{(1,0)}$.dirección=2 AND $C_{(1,0)}$.destino-0=0 para ese peatón
14. $C_{(1,0)}$.dirección=2 AND $C_{(1,0)}$.destino-0≠0 para ese peatón AND $C_{(0,-1)}$.dirección>0 AND $C_{(0,1)}$.dirección>0.
15. $C_{(1,0)}$.dirección=3 AND $C_{(1,0)}$.destino-0=0 para ese peatón AND rand()=(0,0) AND $C_{(2,1)}$.dirección>0 AND $C_{(1,1)}$.dirección>0 AND $C_{(0,1)}$.dirección>0.
16. $C_{(1,0)}$.dirección=3 AND $C_{(1,0)}$.destino-0<0 para ese peatón AND $C_{(1,1)}$.dirección>0 AND $C_{(0,1)}$.dirección>0.
17. $C_{(1,0)}$.dirección=4 AND $C_{(1,0)}$.destino-0=0 para ese peatón AND rand()=(0,0) AND $C_{(2,-1)}$.dirección>0 AND $C_{(1,-1)}$.dirección>0 AND $C_{(0,-1)}$.dirección>0.
18. $C_{(1,0)}$.dirección=4 AND $C_{(1,0)}$.destino-0<0 para ese peatón AND $C_{(1,-1)}$.dirección>0 AND $C_{(0,-1)}$.dirección>0.

Estos casos pueden ser construidos para el resto de las celdas adyacente a la origen y su resultado es similar al presentado, teniendo en cuenta pequeños cambios que dependen del vecino particular.

NingúnAdyacenteOcupadoAvanza. Para verificar que esto se cumpla se debe evaluar de las posiciones adyacentes ocupadas cuáles deciden no avanzar hacia la origen. La idea, es semejante al caso anterior, revisando aquellas adyacentes que contienen un peatón que no avanza hacia la origen.

AlgúnAdyacenteOcupadoAvanzaEchando. En este caso alguno de los vecino ocupados cae en la situación que debe avanzar hacia la celda origen a pesar que éste estuviera ocupado.

6. Modelos de control de tráfico utilizando Autómatas Celulares

El tráfico de vehículos constituye un aspecto importante de la vida actual, pues permite la movilidad de gente y productos que son elementos claves para la economía de un país. En zonas industrializadas, el transporte absorbe gran parte del presupuesto. Si los vehículos se retrasan por congestionamientos, se pierde dinero. Por otro lado, el tráfico debe ser controlado, redirigido y optimizado por estar en aumento continuo y por ser una de las fuentes más grande de polución.

El tráfico tiene características que lo llevan a ser un sistema complejo. Tiene estados estables e inestables, determinísticos, caóticos o comportamiento estocástico con fases de transición;

dimensiones fractales y criticidad auto organizada. Para manejar este tipo de sistemas, la simulación juega un rol importante pues sirve como base para investigar diferentes escenarios de estrategias de control de tráfico. Los modelos pueden usarse para evaluar la influencia de cambios en las normas de tránsito, producir salidas para determinar la polución causada por los vehículos; etc. También pueden utilizarse en tiempo real para tratar con áreas congestionadas, dando la posibilidad de evaluar el impacto de cambios de flujos de tráfico causados por accidentes, trabajos en las calles que anulan carriles de circulación, etc.

Los autómatas celulares han demostrado ser una herramienta útil para la simulación del tráfico, puesto que logran generar dinámicas de tráfico razonables.

En la sección 6.1 se presenta una taxonomía que permite clasificar modelos para simulación de tráfico. En base a ella, se organiza la sección 6.2, donde se describen y clasifican modelos de autómatas celulares existentes, atacando su definición formal. Esto permitió descubrir áreas del problema modelizables no representadas, cuyo análisis se realiza en la sección 6.3.

6.1. Definición de una Taxonomía

Un aspecto importante de los modelos de simulación de flujo de vehículos es la estructura elegida para representar las calles, autopistas o vías de circulación; ella determina en gran medida el tipo de movimiento que caracterizará a los vehículos. Esto permite distinguir entre modelos simples que sólo representan el flujo de vehículos sobre un único carril, y los más complejos que modelan varios carriles de distinta dirección y con cruces. Estos últimos, de acuerdo a su estructura, pueden hacer que los vehículos tengan un movimiento más completo; modelando por ejemplo, giros y cambios de carril.

Otro aspecto importante que hay que considerar al simular tráfico, son las características específicas que se desean modelar que afectan el movimiento de los vehículos, a parte de la estructura de las calles. Esto puede incluir la representación de diversos tipos de vehículos (autos, camiones, colectivos, taxis, etc.), señales de tránsito, desvíos, choques, etc.

Teniendo en cuenta los aspectos mencionados se definen dos clasificaciones para modelos de simulación de tráfico, en la primera se utiliza como criterio la estructura de las calles y en la segunda las características adicionales que afectan el movimiento de los vehículos. Estas últimas son independientes de la estructura de las calles, es decir, pueden estar presentes o no en cualquiera de las subclases determinadas por la primera; este fue el motivo principal para definir dos clasificaciones en vez de una sola. Por lo tanto, ambas son complementarias y utilizadas en conjunto permiten especificar una mayor cantidad de características del modelo.

En definitiva, las clasificaciones resultantes para analizar modelos existentes de autómatas celulares para simulación del tráfico son definidas como:

1) Clasificación según la estructura de las calles

A. Modelos de tráfico de carril único
   A.1. Modelos de una dimensión (sin cruces)
   A.2. Modelos de dos dimensiones (con cruces)
B. Modelos de tráfico de dos carriles
   B.1. Modelos con una única dirección
      B.1.1. Modelos con reglas simétricas
         B.1.1.1. Modelos sin cruce de carriles
         B.1.1.2. Modelos con cruce de carriles
      B.1.2. Modelos con reglas asimétricas

B.1.2.1. Modelos sin cruce de carriles
B.1.2.2. Modelos con cruce de carriles
B.2. Modelos con dos direcciones
B.2.1. Modelos sin cruces de carriles
B.2.2. Modelos con cruces de carriles
C. Modelos de tráfico de más de dos carriles
C.1. Modelos con una única dirección
C.1.1. Modelos con reglas simétricas
C.1.1.1. Modelos sin cruces de carriles
C.1.1.2. Modelos con cruces de carriles
C.1.2. Modelos con reglas asimétricas
C.1.2.1. Modelos sin cruces de carriles
C.1.2.2. Modelos con cruces de carriles
C.2. Modelos con dos direcciones
C.2.1. Modelos con reglas simétricas
C.2.1.1. Modelos sin cruces de carriles
C.2.1.2. Modelos con cruces de carriles
C.2.2. Modelos con reglas asimétricas
C.2.2.1. Modelos sin cruces de carriles
C.2.2.2. Modelos con cruces de carriles

En los modelos de carril único el tráfico se representa como una única línea con flujo de vehículos. Luego pueden permitir o no que estos carriles independientes se crucen para formar las esquinas, constituyendo los modelos de dos dimensiones (con cruces) y de una dimensión (sin cruces), respectivamente.

Los modelos de tráfico de dos carriles se representan como dos líneas con flujo de vehículos paralelas. A su vez estos modelos pueden permitir que los dos carriles tengan la misma o distinta dirección de circulación. En el primer caso, el movimiento de cambio de carril puede definirse con reglas simétricas o asimétricas. Las reglas son simétricas si las condiciones para que los vehículos pasen de un carril a otro, son las mismas; caso contrario son asimétricas y se utilizan cuando por ejemplo un carril se utiliza sólo para adelantarse.

Los modelos de tráfico de más de dos carriles se representan como más de dos líneas con flujo de vehículos paralelas. Estos modelos, por lo general, son extensiones de los de dos carriles y se pueden clasificar considerando los mismo aspectos. La única diferencia es que dentro de los modelos de 2 direcciones (C.2), existe también la posibilidad de definir reglas simétricas o asimétricas para los carriles de igual dirección.

Cada modelo de la clasificación anterior, a su vez puede representar algunas de las siguientes características, o bien, una combinación de ellas,

2) Clasificación según características específicas a modelar

I. Vehículos especiales: en este caso se modelan vehículos con alguna característica o comportamiento diferente. Entre ellos se pueden considerar los siguientes:
I.1. Autos
I.2. Camiones
I.3. Colectivos
I.4. Ambulancias
I.5. Taxis
I.6. Bicicletas
I.7. Motos
I.8. Vehículos con ruta prefijada

II. Controles sobre el tráfico: en este caso los vehículos deben respetar las normas de tránsito incluidas en el modelo, que pueden ser:
  II.1. Semáforos (con/sin flecha de giro)
  II.2. Señales (PARE, Escuela, No Adelantarse, Ceda el Paso, Velocidad Máxima, Velocidad Mínima, Cruce de Peatones, Zona de Animales Sueltos, etc.)
  II.3. Cruces de trenes
    II.3.1. Con barrera
    II.3.2. Sin barrera
    II.3.3. Sólo el tren, sin cruce
  II.4. Carriles con prioridades fijas, como por ejemplo, los autos que llegan por la derecha a un cruce tienen prioridad sobre los otros
  II.5. Carriles especiales (por ejemplo, exclusivos para taxis y colectivos)
  II.6. Bocacalles
  II.7. Elevaciones transversales (lomas de burro)
  II.8. Depresiones transversales (badén)
  II.9. Irregularidad continua (serrucho)
III. Comportamientos anómalos de los vehículos: que pueden incluir,
  III.1. Autos con movimiento erráticos (zig zag)
  III.2. Autos que circulan por el medio entre dos carriles
  III.3. Autos maniobrando para estacionar
  III.4. Autos detenidos
  III.5. Autos circulando marcha atrás
IV. Obstáculos en la calle:
  IV.1. Obras (hombres trabajando)
  IV.2. Baches
  IV.3. Choques y otros accidentes
  IV.4. Autos mal estacionados
    IV.4.1. En doble mano
    IV.4.2. Sobre la mano izquierda
  IV.5. Peatones:
    IV.5.1. Descuidados: cruzan sin mirar si se acercan autos
    IV.5.2. Precavidos: cruzan prestando atención al tráfico (considerando autos que doblan o autos que siguen derecho, con o sin semáforo)

6.2. Clasificación de modelos según la taxonomía

En esta sección se presentan algunos modelos existentes de autómatas celulares para simulación de tráfico de vehículos, analizando su clasificación según la taxonomía definida, y planteando una posible especificación para cada uno siguiendo la formalización de la sección 4. Aquí se ordenan los modelos según su estructura de calles (primera clasificación) y a su vez se los separa en *modelos simples* (sólo representan el movimiento de autos, que se desplazan mientras haya lugar) o modelos con alguna caracterísitica de la segunda clasificación (con semáforos, camiones, etc.).

A. **Modelos de tráfico de carril único**

En general, los modelos que consideran un único carril se definen como base para ser extendidos a varios carriles, puesto que estos últimos tienden a ser más significativos por estar más próximos a la realidad. En esta sección se estudian algunos de estos modelos existentes.

A.1. **Modelos de una dimensión (sin cruces)**

En estos modelos el tráfico se representa sobre un carril continuo que no tiene cruces ni intersecciones. Esto permite que las reglas del modelo sean bastante simples pues no considera aspectos como prioridad de acceso al cruce, maniobras para pasar a otro vehículo, etc.

A.1.i. **Modelos simples**

En [CMB93] se plantea un modelo para el tráfico sobre un carril sin cruces utilizando un autómata celular unidimensional. El estado de cada celda está dado por la presencia o ausencia de un auto. En cada paso de tiempo se debe considerar si el auto que está ocupando una celda se quedará en ella detenido o se desplazará hacia la de adelante; para tomar esa decisión se usan las reglas del modelo sobre el vecindario, que está formado por la celda de adelante y la de atrás. Un auto se mueve hacia adelante, siempre y cuando tenga un espacio libre; caso contrario permanece en su posición.

Este modelo se puede representar como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^{+} >$$

S = { 0, 1 }

Donde,

0 representa la celda vacía,

1 representa que la celda tiene un vehículo.

n = 1

$\boldsymbol{\eta}$ = 3

N = { 0, -1, 1 }

-1	0	1

Figura 71 - Vecindario de la celda origen

La función $\boldsymbol{\tau}$ se define como:

Nuevo Estado	Estado del vecindario
0	(C_{-1}= 0 AND C_0= 0) OR (C_0= 1 AND C_1= 0)
1	(C_{-1}= 1 AND C_0= 0) OR (C_0= 1 AND C_1= 1)

Otro modelo dentro de esta categoría se presenta en [NS92]. A diferencia del anterior, se considera que los vehículos pueden circular a distintas velocidades y en un mismo paso de tiempo pueden atravesar varias celdas. Cada celda puede estar vacía u ocupada por un auto, quien a su vez tiene una velocidad asociada que representa el número de posiciones que avanza en cada actualización.

Notación 7:

De aquí en adelante **x** denotará a la posición actual del vehículo, **v** a su velocidad (número entero entre 0 y v_{max}), $\mathbf{v_{max}}$ a la velocidad máxima que puede alcanzar, y **gap** a la cantidad de posiciones vacías entre el vehículo y el auto de adelante.

En este modelo se definen reglas de comportamiento que se aplicadan simultáneamente a todos los vehículos en 2 pasos. El primero, se denomina *actualización de velocidades* y se utilizan 3 reglas:

(1) Si $v < v_{max}$ entonces $v = v + 1$ (el auto acelera)
(2) Si $v > gap$ entonces $v = gap$ (el auto frena un poco para no chocar con el auto de adelante, ubicándose en la posición inmediatamente detrás del otro auto)
(3) Si $v > 0$ entonces $v = v - 1$ con probabilidad p_{brake} (un auto frena en forma no determinística)

El segundo paso se denomina *actualización de la posición* y se utiliza la regla:

(4) $x = x + v$ (actualiza la posición de acuerdo a la velocidad calculada en el paso anterior)

Este modelo se puede representar como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

$\mathbf{S} = \{ -1, 0, 1, 2, ..., v_{max} \}$

Donde,

-1 representa la celda vacía,
0 representa que la celda tiene un vehículo con velocidad 0,
1 representa que la celda tiene un vehículo con velocidad 1, etc; y
v_{max} representa la velocidad máxima que puede alcanzar cualquier vehículo;

$\mathbf{n} = 1$

$\boldsymbol{\eta} = 2*v_{max} + 1$

$\mathbf{N} = \{ - v_{max}, -v_{max}+1, v_{max}+2, ..., -2, -1, 0, 1, 2, ..., v_{max}-1, v_{max} \}$

$-v_{max}$	...	-2	-1	0	1	2	...	v_{max}

Figura 72 - Vecindario de la celda origen

Notación 8:

De aquí en adelante **brake** representa una función booleana que devuelve verdadero con probabilidad p_{brake} (p_{brake} es la probabilidad con la que un vehículo frena sin ninguna razón); y en ese caso el auto disminuye su velocidad.
Además **gap(i)** indica la cantidad de posiciones vacías entre la celda i y su primer vecina hacia adelante ocupada por un vehículo, es decir, aquella cuyo estado sea mayor que –1. Por ejemplo, si C_{i+1}=-1 y C_{i+2} =-1 pero C_{i+3}=4 entonces gap(i) vale 2.

Para la función τ se describen 2 conjuntos de reglas que se aplican en dos pasos, en el primero se actualiza la velocidad de cada vehículo y en el segundo su posición de acuerdo a la nueva velocidad.

Reglas para actualizar la velocidad (paso 1):

Nuevo Estado	Estado del vecindario
-1	$C_0 = -1$
gap(0)	[$C_0 > -1$ AND $C_0 < v_{max}$ AND $C_0 + 1 > gap(0)$ AND $gap(0) > 0$ AND NOT(brake)] OR [$C_0 > -1$ AND $C_0 = v_{max}$ AND $C_0 > gap(0)$ AND $gap(0) > 0$ AND NOT(brake)] OR [$C_0 > -1$ AND $C_0 < v_{max}$ AND $C_0 + 1 > gap(0)$ AND $gap(0) = 0$] OR [$C_0 > -1$ AND $C_0 = v_{max}$ AND $C_0 > gap(0)$ AND $gap(0) = 0$]
gap(0)-1	[$C_0 > -1$ AND $C_0 < v_{max}$ AND $C_0 + 1 > gap(0)$ AND $gap(0) > 0$ AND brake] OR [$C_0 > -1$ AND $C_0 = v_{max}$ AND $C_0 > gap(0)$ AND $gap(0) > 0$ AND brake]
C_0	$C_0 > -1$ AND $C_0 + 1 \leq gap(0)$ AND $C_0 < v_{max}$ AND brake
$C_0 + 1$	$C_0 > -1$ AND $C_0 + 1 \leq gap(0)$ AND $C_0 < v_{max}$ AND NOT(brake)
v_{max}	$C_0 = v_{max}$ AND $C_0 \leq gap(0)$ AND NOT(brake)
v_{max}-1	$C_0 = v_{max}$ AND $C_0 \leq gap(0)$ AND brake

Reglas para actualizar la posición de los vehículos (paso 2):

Nuevo Estado	Estado del vecindario
C_0	$C_0 = 0$
C_k	k es el vecino hacia atrás más cercano ocupado AND $C_k > -1$ AND $C_k + k = 0$ AND $C_0 = -1$
-1	[k es el vecino hacia atrás más cercano ocupado AND $C_k > -1$ AND ($C_k + k < 0$ OR $C_k + k > 0$) AND $C_0 = -1$] OR (Todas las posiciones hacia atrás vacías AND $C_0 = -1$) OR [$C_0 > 0$]

Donde,
"El vecino hacia atrás más cercano ocupado", es aquel vecino ocupado por un auto con el número de índice máximo, menor a cero. Por ejemplo, si $C_{-1}=-1$ y $C_{-2}=-1$ pero $C_{-3}=4$ entonces el vecino hacia atrás más cercano ocupado es -3.
"Todas las posiciones hacia atrás vacías", equivale a la siguiente conjunción:

$$\bigwedge_{i \in \{-v\max,\ldots,-1\}} C_i = -1$$

A.1.ii. **Modelos con semáforos**

Otro modelo de una dimensión se propone en [SN97], que además representa semáforos que controlan el flujo del tráfico. Allí se presentan varias formas de incluir esta característica sobre un modelo simple. Se describe un carril formado por nodos y enlaces, donde un *enlace* es un segmento de una calle con dirección y los *nodos* representan las intersecciones o cruces. Cada enlace puede ser definido por el nodo de entrada y el de salida. Los vehículos se mueven por algún enlace y pasan a otro siguiendo una ley estocástica simple, que representa la presencia de un semáforo.

Cada enlace es modelado por un *autómata celular* con las características y reglas de movimiento propuestas en [NS92] (modelo anterior). La velocidad de un vehículo se actualiza

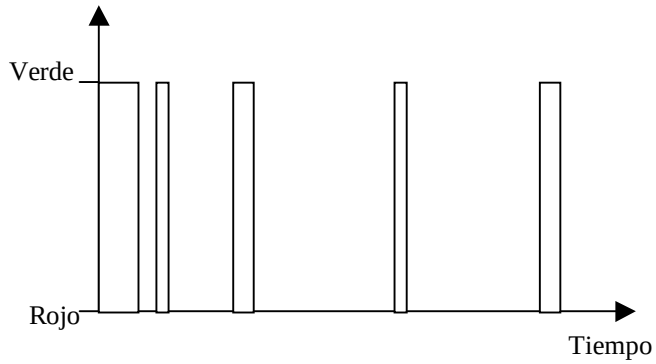
según las reglas del modelo para un carril hasta que llega a una intersección. Si el vehículo puede atravesar la intersección, de acuerdo al movimiento de avance, se chequea el semáforo. Si está en verde, entonces el vehículo mantiene su velocidad y pasa al otro enlace. Caso contrario (está en rojo), se coloca un "vehículo virtual" en la primera posición del otro enlace forzando a que el auto frene detrás. Cuando el semáforo pasa a verde, el auto virtual se elimina.

En [SN97] se proponen tres formas distintas para modelar la presencia de un semáforo:

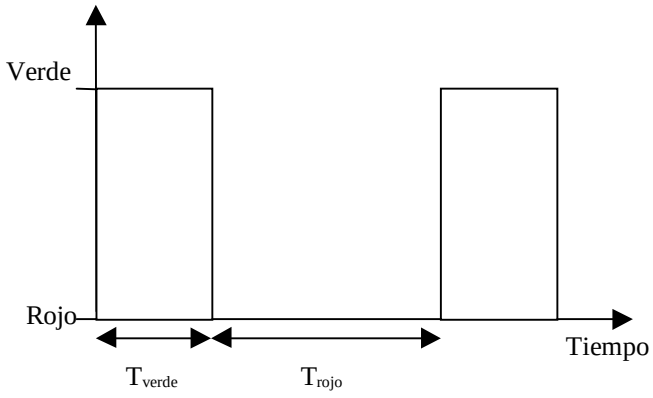
1. Semáforos aleatorios: el semáforo pasa de rojo a verde con probabilidad p_{trans} y la fracción de tiempo (f_{verde}) que permanece en verde es exactamente p_{trans}. Para este caso se tendría la siguiente regla de actualización de la velocidad:
   IF (rand < p_{trans}) THEN actualización normal
   ELSE gap = distancia desde el vehículo hasta la intersección.
2. Semáforos con distribución normal: la fracción de tiempo en que el semáforo se mantiene en verde está dada por $f_{verde} = T_{verde}/(T_{verde}+T_{rojo})$.
3. Semáforos de Dirac: el objetivo es setear el semáforo a verde o rojo por una sólo unidad de tiempo, equitativamente espaciados en un ciclo. La fracción de tiempo en que el semáforo se mantiene en verde está dada por $f_{verde}=1/(1+T_{rojo})$ con $T_{rojo}\geq 1$ (y $T_{verde}=1$); ó $1 – (1/(1+T_{verde}))$ con $T_{rojo}\geq 1$.

En las siguientes figuras se muestran las distintas formas de distribuir la duración del semáforo en verde.

Semáforo Aleatorios



Semáforo con distribución Normal



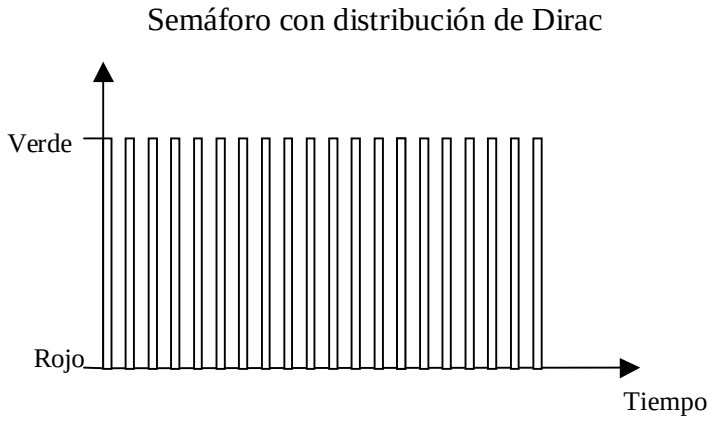


Figura 73 – Diversos Semáforos

Este modelo se puede representar como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^{+} >$$

S = { -1, 0, 1, 2, ..., v_{max} }

Donde,

-1 representa la celda vacía,
0 representa que la celda tiene un vehículo con velocidad 0,
1 representa que la celda tiene un vehículo con velocidad 1, etc.

n = 1

$\boldsymbol{\eta}$ = $2*v_{max} + 1$

N = { - v_{max}, $-v_{max}+1$, $v_{max}+2$,..., -2, -1, 0, 1, 2, ..., $v_{max}-1$, v_{max} }

$-v_{max}$	...	-2	-1	0	1	2	...	v_{max}

Figura 74 - Vecindario de la celda origen

La función $\boldsymbol{\tau}$ se calcula en dos pasos semejantes a los descriptos para el modelo de [NS92]. El primero (cálculo de la velocidad) se define como:

Si ($d_{intersec} > v_{max}$ OR gap$\leq d_{intersec}$ OR semáforo(t)=verde)
  Entonces aplicar las reglas del paso 1 tal como fueron definidas para [NS92]
Sino // $d_{intersec} \leq v_{max}$ AND gap$> d_{intersec}$ AND semáforo(t)=rojo //
  $C_0 = d_{intersec}$

Donde, cada celda conoce el valor de $d_{intersec}$, que indica a qué distancia se encuentra la misma de la intersección hacia adelante más cercana; y además las próximas al semáforo, pueden monitorear su color, usando una función (semáforo()). Para el caso de los semáforos aleatorios, esta función devuelve un número que indica rojo o verde, al azar. Para los otros dos tipos de semáforos, esta función calcula el color según las distribuciones correspondientes.

Para el segundo paso, actualización de posiciones, la función $\boldsymbol{\tau}$ queda definida exactamente igual que como se hizo para [NS92].

A.1.iii. **Modelos con vehículos especiales**

El modelo de [NS92] es ampliado agregando la siguiente condición: los autos pueden tener distinta v_{max}; pero para cada v_{max}, p_{brake} tiene el mismo valor. De esta forma aquellos vehículos que tienen una velocidad máxima baja, pueden representar a camiones o algún otro medio de transporte lento.

Este modelo se representa en forma similar al de [NS92], pero difiere en que es necesario incluir dentro del estado de la celda la velocidad máxima que puede alcanzar el vehículo que contiene. Luego cada celda debe conocer la probabilidad de frenar, p_{brake}, asociada a cada v_{max}, esta información es constante durante toda la simulación, por lo tanto no es necesario incluir ese valor como parte del estado. En definitiva, se obtiene:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

$\mathbf{S} = \{ (v_{actual}, v_{max}) / v_{actual} \in \{-1, 0, 1, 2, ..., v_{max} \}, v_{max} \in \{1, 2, ...,Max \} \}$

Donde,

$v_{actual} = -1$ representa la celda vacía,

$v_{actual} = 0$ representa que la celda tiene un vehículo con velocidad 0,

$v_{actual} = 1$ representa que la celda tiene un vehículo con velocidad 1, etc.

Max es una constante que se usa como parámetro de la simulación para indicar la mayor velocidad máxima posible.

$\mathbf{n} = 1$

$\boldsymbol{\eta} = 2*Max + 1$

$\mathbf{N} = \{ -Max , - Max +1, -Max +2,..., -2, -1, 0, 1, 2, ..., Max -1, Max \}$

-Max	...	-2	-1	0	1	2	...	Max

Figura 75 - Vecindario de la celda origen

La función $\boldsymbol{\tau}$ se aplica en dos pasos, en el primero se actualiza la velocidad de cada vehículo y en el segundo su posición de acuerdo a la nueva velocidad.

Reglas para actualizar la velocidad (paso 1):

Nuevo Estado	Estado del vecindario
(-1,0)	$C_0.v_{actual} = -1$
(gap(0),$C_0.v_{max}$)	[$C_0.v_{actual} > -1$ AND $C_0.v_{actual} < C_0.v_{max}$ AND $C_0.v_{actual} +1 >$ gap(0) AND gap(0) > 0 AND NOT(brake)] OR [$C_0.v_{actual} > -1$ AND $C_0.v_{actual} = C_0.v_{max}$ AND $C_0.v_{actual} >$ gap(0) AND gap(0) > 0 AND NOT(brake)] OR [$C_0.v_{actual} > -1$ AND $C_0.v_{actual} < C_0.v_{max}$ AND $C_0.v_{actual} +1 >$ gap(0) AND gap(0) = 0] OR [$C_0.v_{actual} > -1$ AND $C_0.v_{actual} = C_0.v_{max}$ AND $C_0.v_{actual} >$ gap(0) AND gap(0) = 0]
(gap(0)-1,$C_0.v_{max}$)	[$C_0.v_{actual} > -1$ AND $C_0.v_{actual} < C_0.v_{max}$ AND $C_0.v_{actual} +1 >$ gap(0) AND gap(0) > 0 AND brake] OR [$C_0.v_{actual} > -1$ AND $C_0.v_{actual} = C_0.v_{max}$ AND $C_0.v_{actual} >$ gap(0) AND gap(0) > 0 AND brake]

$(C_0.v_{actual}, C_0.v_{max})$	$C_0.v_{actual} > -1$ AND $C_0.v_{actual} + 1 \leq gap(0)$ AND $C_0.v_{actual} < C_0.v_{max}$ AND brake
$(C_0.v_{actual} + 1, C_0.v_{max})$	$C_0.v_{actual} > -1$ AND $C_0.v_{actual} + 1 \leq gap(0)$ AND $C_0.v_{actual} < C_0.v_{max}$ AND NOT(brake)
$(C_0.v_{max}, C_0.v_{max})$	$C_0.v_{actual} = C_0.v_{max}$ AND $C_0.v_{actual} \leq gap(0)$ AND NOT(brake)
$(C_0.v_{max}-1, C_0.v_{max})$	$C_0.v_{actual} = C_0.v_{max}$ AND $C_0.v_{actual} \leq gap(0)$ AND brake

Donde,
brake, sigue la notación 4 pero con la diferencia que recibe como parámetro la velocidad máxima que puede alcanzar el vehículo pues la probabilidad que el auto frene depende de ella.

Reglas para actualizar la posición de los vehículos (paso 2):

Nuevo Estado	Estado del vecindario
$(C_0.v_{actual}, C_0.v_{max})$	$C_0 = 0$
$(C_k.v_{actual}, C_k.v_{max})$	k es el vecino hacia atrás más cercano ocupado AND $C_k.v_{actual} > -1$ AND $C_k.v_{actual} + k = 0$ AND $C_0.v_{actual} = -1$
(-1,0)	[k es el vecino hacia atrás más cercano ocupado AND $C_k.v_{actual} > -1$ AND ($C_k.v_{actual} + k < 0$ OR $C_k.v_{actual} + k > 0$) AND $C_0.v_{actual} = -1$] OR (Todas las posiciones hacia atrás vacías AND $C_k.v_{actual} = -1$) OR [$C_0.v_{actual} > 0$]

Donde,
"El vecino hacia atrás más cercano ocupado", es aquel vecino ocupado por un auto con el número de índice máximo, menor a cero. Por ejemplo, si $C_{-1}.v_{actual} = -1$ y $C_{-2}.v_{actual} = -1$ pero $C_{-3}.v_{actual} = 4$ entonces el vecino hacia atrás más cercano ocupado es -3.
"Todas las posiciones hacia atrás vacías", equivale a la siguiente conjunción:

$$\bigwedge_{i \in \{-C_i.v_{max}, \ldots, -1\}} C_i.v_{actual} = -1$$

A.2. **Modelos de dos dimensiones (con cruces)**

Estos modelos representan calles de un sólo carril que tienen cruces entre sí. De esta forma es necesario definir reglas que además de modelar la circulación de vehículos por un sólo carril, tengan en cuenta un comportamiento distinto al llegar al cruce. Allí los vehículos pueden, por ejemplo, girar o no y competir por acceso a la intersección; todos estos aspectos deben ser definidos en el modelo.

A.2.i. **Modelos simples**

Una alternativa para obtener un modelo de dos dimensiones es extender alguno de los planteados para una, incorporando incrementalmente otros aspectos del tráfico real. Este es el caso del modelo planteado en [CMB93], que primero propone un autómata de una dimensión (presentado en la sección A.1) y luego lo extiende agregando *cruces* o intersecciones entre calles. Para eso, se define un nuevo tipo de celda que son las de cruce, cuyo comportamiento se describe con los siguientes 4 casos:
(El símbolo "  " representa una celda de cruce)

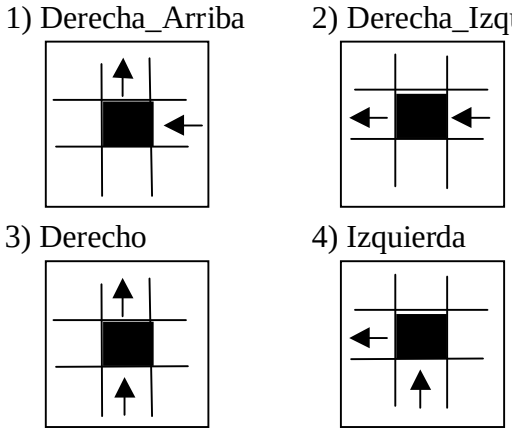


Figura 76 – Cruces

Como muestra la Figura 76, cada celda de cruce tiene 2 entradas y 2 salidas; además pueden ser ocupadas por un sólo auto, que permanece en ella si no hay un lugar vacío al que pueda salir. Los estados posibles de una celda de cruce son: vacía, yendo a Izquierda o yendo hacia Adelante. Para determinar cuál de estos dos últimos corresponde al cruce con un auto en él, se efectúa una elección no determinística de la dirección con la que saldrá del mismo. Otro aspecto que hay que considerar es qué sucede cuando dos vehículos, uno por cada entrada, intenta acceder al cruce. La decisión tomada frente a este conflicto de acceso al cruce es que el auto que llega por la derecha tiene prioridad de avanzar.

En definitiva, la descripción de una celda común (sección A.1) junto con la descripción de una celda de cruce permite definir cualquier sistema de calles planar sin semáforos.

Para la definición del autómata celular que represente este modelo hay que tener en cuenta la existencia de celdas con distinto comportamiento y vecindad. Por un lado, están las *celdas "normales"*, que son aquellas que no son cruces ni son adyacentes a ellos; cuyo comportamiento y vecindad fuera definido en la sección A.1, para el modelo de una dimensión de [CMB93]. Luego existen las *celdas de cruce*, las *celdas de entrada* a un cruce y las *celdas de salida* de un cruce. Aquí sólo se definirá el comportamiento de estos tres últimos tipos, representando el modelo como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\mathbf{Z}_0^+ >$$

S = { 0, 1, 2, 3 }

Donde,

0 representa la celda (normal, de cruce, de entrada o de salida) vacía,
1 representa que la celda de entrada, de salida o normal tiene un vehículo
2 representa que la celda de cruce tiene un vehículo yendo a Izquierda,
3 representa que la celda de cruce tiene un vehículo yendo a Adelante.

n = 2

Celdas de Cruce

η = 5

N = {(0,0); (1,0); (0,-1); (0,1); (-1,0) }

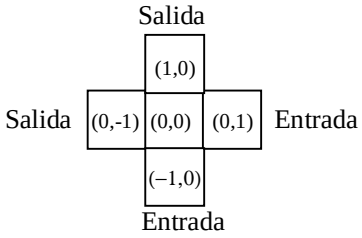


Figura 77 - Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)} = 0$ AND $C_{(0,1)} = 0$ AND $C_{(-1,0)} = 0$) OR ($C_{(0,0)} = 2$ AND $C_{(0,-1)} = 0$) OR ($C_{(0,0)} = 3$ AND $C_{(1,0)} = 0$)
2	($C_{(0,0)} = 2$ AND $C_{(0,-1)} = 1$) OR ($C_{(0,0)} = 0$ AND rand=2 AND $C_{(0,1)} = 1$) OR ($C_{(0,0)} = 0$ AND rand=2 AND $C_{(0,1)} = 0$ AND $C_{(-1,0)} = 1$)
3	($C_{(0,0)} = 3$ AND $C_{(1,0)} = 1$) OR ($C_{(0,0)} = 0$ AND rand=3 AND $C_{(0,1)} = 1$) OR ($C_{(0,0)} = 0$ AND rand=3 AND $C_{(0,1)} = 0$ AND $C_{(-1,0)} = 1$)

Donde, rand puede valer 2 (hacia izquierda) ó 3 (hacia arriba) y define la dirección de salida del vehículo que ingresó al cruce. Esta función elige alguna de las dos posibilidades al azar y su presencia es necesaria para que los vehículos decidan si girar o no al llegar al cruce.
Cabe destacar que estas reglas le dan la prioridad de acceso al cruce a los vehículos que ingresan por la derecha, es decir desde la posición (0,1).

Celdas de Entrada a un cruce

$\eta = 4$

Las celdas de entrada por derecha a un cruce tienen distinto comportamiento que las de entrada por abajo. Esto se debe a que las primeras tienen prioridad de acceso.

Celdas de entrada por derecha

N = { (0,0); (-1,-1); (0,-1); (0,1)}

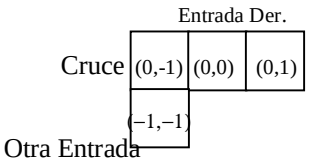


Figura 78 - Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)} = 0$ AND $C_{(0,1)} = 0$) OR ($C_{(0,0)} = 1$ AND $C_{(0,-1)} = 0$)
1	($C_{(0,0)} = 0$ AND $C_{(0,1)} = 1$) OR ($C_{(0,0)} = 1$ AND $C_{(0,-1)} > 0$)

Celdas de entrada por abajo

N = { (0,0); (-1,0); (1,0); (1,1)}

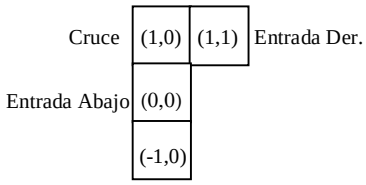


Figura 79 - Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)} = 0$ AND $C_{(-1,0)} = 0$) OR ($C_{(0,0)} = 1$ AND $C_{(1,0)} = 0$ AND $C_{(1,1)} = 0$)
1	($C_{(0,0)} = 0$ AND $C_{(-1,0)} = 1$) OR ($C_{(0,0)} = 1$ AND ($C_{(1,0)} > 0$ OR $C_{(1,1)} = 1$))

Celdas de Salida de un cruce

$\eta = 3$

Las celdas de salida por izquierda de un cruce tienen distinto comportamiento que las de salida por arriba.

Celdas de salida por izquierda

N = { (0,0); (0,-1); (0,1) }

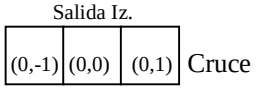


Figura 80 - Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)} = 0$ AND $C_{(0,1)} = 0$) OR ($C_{(0,0)} = 1$ AND $C_{(0,-1)} = 0$)
1	($C_{(0,0)} = 0$ AND $C_{(0,1)} = 2$) OR ($C_{(0,0)} = 1$ AND $C_{(0,-1)} = 1$)

Celdas de salida por arriba

N = { (0,0); (1,0); (-1,0) }

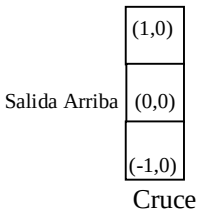


Figura 81 - Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)} = 0$ AND $C_{(-1,0)} = 0$) OR ($C_{(0,0)} = 1$ AND $C_{(1,0)} = 0$)
1	($C_{(0,0)} = 0$ AND $C_{(-1,0)} = 3$) OR ($C_{(0,0)} = 1$ AND $C_{(1,0)} = 1$)

A.2.ii. **Modelos con semáforos**

En [T96] se plantea un modelo en dos dimensiones que tiene en cuenta la presencia de semáforos para resolver el acceso a posiciones del cruce de dos calles. Para ello se utiliza una grilla de NxN, donde circulan los autos con una dinámica determinística, y cuyos bordes no se conectan entre sí (open boundaries), es decir las celdas de los bordes tienen un comportamiento diferente a las demás. Los autos son inyectados probabilísticamente por el borde izquierdo y el de abajo, y se mueven determinísticamente hacia la derecha y arriba, respectivamente.

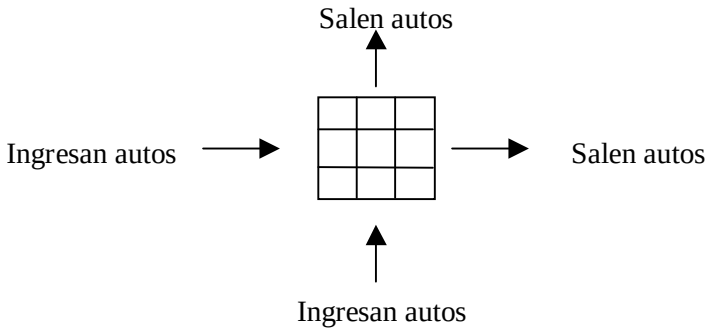


Figura 82 – Bordes del modelo

Cada celda del autómata puede estar vacía u ocupada por un auto. Los autos tienen una dirección definida en el momento en que ingresan a la grilla, que puede ser hacia arriba o hacia la derecha y se pueden mover de a un paso por vez, siempre y cuando la celda a la que quieren ir esté vacía.

Para modelar la presencia de *semáforos* se impone una restricción sobre el comportamiento de los autos; de manera que los autos que se dirigen hacia arriba, sólo se pueden mover en pasos de tiempo pares, y los autos que se dirigen hacia la derecha, sólo se pueden mover en los impares.

Para plantear las reglas de la evolución del estado del autómata, se utilizan 2 vectores, μ y ν, que se definen de la siguiente forma:

- $\mu_{\vec{r}}(t) = 1$ si el auto de la posición $\vec{r} = (i, j)$ se dirige hacia la derecha en el instante t; $\mu_{\vec{r}}(t) = 0$ en otro caso.

- $\nu_{\vec{r}}(t) = 1$ si el auto de la posición $\vec{r} = (i, j)$ se dirige hacia arriba en el instante t; $\nu_{\vec{r}}(t) = 0$ en otro caso.

Se definen tres tipos de celdas con su comportamiento asociado, ellas son las celdas de ingreso de los vehículos (borde izquierdo e inferior de la grilla), las celdas de salida (borde derecho y superior de la grilla) y el resto (celdas de movimiento de los autos). A continuación se presentan primero las reglas de movimiento de los autos, luego las de ingreso a la grilla y por último las de salida.

Las reglas para el *movimiento de los autos* se definen mediante las siguientes ecuaciones:

$$\begin{aligned}\mu_{\vec{r}}(t+1) = {} & \sigma(t)\ \mu_{\vec{r}}(t)\ \{\mu_{\vec{r}+\vec{x}}(t) + \nu_{\vec{r}+\vec{x}}(t)\} \\ & + \sigma(t)\ \{1 - \mu_{\vec{r}}(t)\}\{1 - \nu_{\vec{r}}(t)\}\ \mu_{\vec{r}-\vec{x}}(t) \\ & + \{1 - \sigma(t)\}\ \mu_{\vec{r}}(t) \qquad (1)\end{aligned}$$

$$\begin{aligned}\nu_{\vec{r}}(t+1) = {} & \{1-\sigma(t)\}\ \nu_{\vec{r}}(t)\ \{\nu_{\vec{r}+\vec{y}}(t) + \mu_{\vec{r}+\vec{y}}(t)\} \\ & + \{1-\sigma(t)\}\ \{1 - \mu_{\vec{r}}(t)\}\{1 - \nu_{\vec{r}}(t)\}\ \mu_{\vec{r}-\vec{y}}(t) \\ & + \sigma(t)\ \nu_{\vec{r}}(t) \qquad (2)\end{aligned}$$

En las reglas anteriores se utilizan los vectores $\vec{x}$ e $\vec{y}$ para denotar los valores $\vec{x} = (0,1)$ e $\vec{y} = (1,0)$. Además, la función binaria σ(t) = t mod 2 representa el control realizado por el semáforo, que separa los movimientos de los vehículos con distintas direcciones en ciclos pares e impares. Las ecuaciones verifican la condición $\mu_{\vec{r}}(t)\ \nu_{\vec{r}}(t) = 0$, para evitar que una posición sea ocupada por un auto que va en dirección hacia arriba y otro con dirección hacia la derecha simultáneamente.
El primer término de la ecuación (1), $\sigma(t)\ \mu_{\vec{r}}(t)\ \{\mu_{\vec{r}+\vec{x}}(t) + \nu_{\vec{r}+\vec{x}}(t)\}$, denota que un auto con dirección hacia la derecha permanece en la posición $\vec{r}$ si el lugar adyacente derecho está ocupado por un auto que va hacia la derecha o hacia arriba. El segundo término, $\sigma(t)\ \{1 - \mu_{\vec{r}}(t)\}\{1 - \nu_{\vec{r}}(t)\}\ \mu_{\vec{r}-\vec{x}}(t)$, denota el ingreso de un auto con dirección hacia la derecha desde el lugar adyacente izquierdo. En él se verifica que en la posición $\vec{r}$ no haya ningún auto. Y por último el tercer término, $\{1 - \sigma(t)\}\ \mu_{\vec{r}}(t)$, muestra que un auto con dirección hacia la derecha no puede moverse en los ciclos de reloj impares. Una interpretación similar corresponde a los términos de la ecuación (2) .
Las reglas para el *ingreso de los autos* por las celdas de los bordes izquierdo y de abajo son una variante de las anteriores, que agregan un nuevo auto con probabilidad p si la celda está vacía. La inyección de autos con dirección hacia la derecha se expresa con la siguiente ecuación:

$$\begin{aligned}\mu_{\vec{r}}(t+1) = {} & \sigma(t)\ \mu_{\vec{r}}(t)\ \{\mu_{\vec{r}+\vec{x}}(t) + \nu_{\vec{r}+\vec{x}}(t)\} \\ & + \sigma(t)\ \{1 - \mu_{\vec{r}}(t)\}\{1 - \nu_{\vec{r}}(t)\}\ f(p) \\ & + \{1 - \sigma(t)\}\ \mu_{\vec{r}}(t) \qquad (3)\end{aligned}$$

Donde,
- $\vec{r} = (1, j)$, con $(1 \le j \le N)$. La inyección de autos dirigidos hacia la derecha se realiza en el borde izquierdo de la grilla.
- La inyección es probabilística utilizando la función f(p) = {0,1} que devuelve 1 con probabilidad p.

La inyección de autos con dirección hacia arriba se expresa con la siguiente ecuación:

$$\nu_{\vec{r}}(t+1) = \{1-\sigma(t)\}\ \nu_{\vec{r}}(t)\ \{\nu_{\vec{r}+\vec{y}}(t) + \mu_{\vec{r}+\vec{y}}(t)\}$$

$$+ \{1-\sigma(t)\}\ \{1 - \mu_{\vec{r}}(t)\}\{1 - \nu_{\vec{r}}(t)\}\ f(p)$$
$$+ \sigma(t)\ \nu_{\vec{r}}(t) \qquad (4)$$

Donde,

- $\vec{r} = (i,1)$, con (1 ≤ i ≤ N). La inyección de autos dirigidos hacia arriba se realiza en el borde inferior de la grilla.
- La inyección es probabilística utilizando la función f(p) = {0,1} que devuelve 1 con probabilidad p.

Por último, se definen las reglas para la *salida de los autos* por las celdas de los bordes derecho y superior. Para el borde derecho se expresa la siguiente ecuación:

$$\mu_{\vec{r}}(t+1) = \sigma(t)\ \{1 - \mu_{\vec{r}}(t)\}\{1 - \nu_{\vec{r}}(t)\}\ \mu_{\vec{r}-\vec{x}}(t)$$
$$+ \{1 - \sigma(t)\}\ \mu_{\vec{r}}(t) \qquad (5)$$

Donde,

- $\vec{r} = (N, j)$, con (1 ≤ j ≤ N). La salida de autos dirigidos hacia la derecha se realiza en el borde derecho de la grilla.

La salida de autos por el borde superior se expresa con la siguiente ecuación:

$$\nu_{\vec{r}}(t+1) = \{1-\sigma(t)\}\ \{1 - \mu_{\vec{r}}(t)\}\{1 - \nu_{\vec{r}}(t)\}\ \nu_{\vec{r}-\vec{y}}(t)$$
$$+ \sigma(t)\ \nu_{\vec{r}}(t) \qquad (6)$$

Donde,

- $\vec{r} = (i, N)$, con (1 ≤ i ≤ N). La salida de autos dirigidos hacia la derecha se realiza en el borde derecho de la grilla.

En resumen, en t = 0 el sistema no tiene autos, éstos se inyectan con las ecuaciones (3) y (4) probabilísticamente y avanzan determinísticamente según las ecuaciones (1) y (2). Si los autos llegan a los bordes del sistema, ellos salen del mismo usando las ecuaciones (5) y (6).

Este modelo se puede definir como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

S = { 0, 1, 2 }

Donde,

0 representa la celda vacía,
1 representa que la celda tiene un vehículo que se dirige hacia la derecha,
2 representa que la celda tiene un vehículo que se dirige hacia arriba.

n = 2

Aquí hay tres tipos de celdas (de ingreso, de salida y el resto) con distinto comportamiento y definición de la vecindad; y se definen separadamente.

<u>Celdas que no están en los bordes de la grilla</u>

η = 5

N = { (0,0); (1,0); (0,-1); (0,1); (-1,0) }

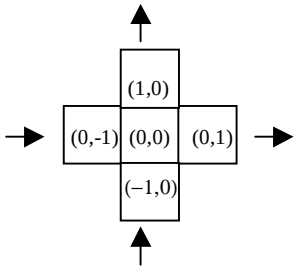


Figura 83 -Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)} = 0$ AND $C_{(0,-1)} = 0$ AND $C_{(-1,0)} = 0$) OR ($C_{(0,0)} = 1$ AND $C_{(0,1)} = 0$ AND σ(time)) OR ($C_{(0,0)} = 0$ AND $C_{(0,-1)} = 1$ AND NOT(σ(time))) OR ($C_{(0,0)} = 2$ AND $C_{(1,0)} = 0$ AND NOT(σ(time))) OR ($C_{(0,0)} = 0$ AND $C_{(-1,0)} = 2$ AND σ(time))
1	($C_{(0,0)} = 0$ AND $C_{(0,-1)} = 1$ AND σ(time)) OR ($C_{(0,0)} = 1$ AND ($C_{(0,1)} = 1$ OR $C_{(0,1)} = 2$) AND σ(time)) OR ($C_{(0,0)} = 1$ AND NOT(σ(time)))
2	($C_{(0,0)} = 0$ AND $C_{(-1,0)} = 2$ AND NOT(σ(time))) OR ($C_{(0,0)} = 2$ AND ($C_{(1,0)} = 1$ OR $C_{(1,0)} = 2$) AND NOT(σ(time))) OR ($C_{(0,0)} = 2$ AND σ(time))

Donde,

time, devuelve el tiempo actual de la simulación.

σ, es una función que se define como: $\sigma(t)$ = t mod 2.

Celdas de ingreso a la grilla

Tanto el borde izquierdo como el inferior, contienen las celdas que permiten el ingreso de nuevos vehículos. Para cada uno de ellos se define una vecindad y comportamiento distinto.

Celdas del borde izquierdo

$\eta = 4$

N = { (0,0); (1,0); (-1,0); (0,1)}

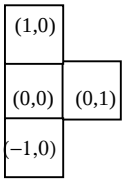


Figura 84 - Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)} = 1$ AND $C_{(0,1)} = 0$ AND σ(time)) OR [$C_{(0,0)} = 0$ AND [(NOT(σ(time)) AND $C_{(-1,0)} \neq 2$) OR (NOT(f(p)) AND σ(time))]] OR ($C_{(0,0)} = 2$ AND $C_{(1,0)} = 0$ AND NOT(σ(time)))
1	($C_{(0,0)} = 0$ AND σ(time) AND f(p)) OR ($C_{(0,0)} = 1$ AND ($C_{(0,1)} = 1$ OR $C_{(0,1)} = 2$)) OR ($C_{(0,0)} = 1$ AND NOT(σ(time)))
2	($C_{(0,0)} = 0$ AND $C_{(-1,0)} = 2$ AND NOT(σ(time))) OR ($C_{(0,0)} = 2$ AND ($C_{(1,0)} = 1$ OR $C_{(1,0)} = 2$)) OR ($C_{(0,0)} = 2$ AND σ(time))

Celdas del borde inferior

$\eta = 4$

N = { (0,0); (1,0); (0,-1); (0,1) }

Figura 85 -Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)} = 2$ AND $C_{(1,0)} = 0$ AND NOT(σ(time))) OR [$C_{(0,0)} = 0$ AND [(σ(time) AND $C_{(0,-1)} \neq 1$) OR (NOT(f(p)) AND NOT(σ(time)))]] OR ($C_{(0,0)} = 1$ AND $C_{(0,1)} = 0$ AND σ(time))
1	($C_{(0,0)} = 0$ AND σ(time) AND $C_{(0,-1)} = 1$) OR ($C_{(0,0)} = 1$ AND ($C_{(0,1)} = 1$ OR $C_{(0,1)} = 2$)) OR ($C_{(0,0)} = 1$ AND NOT(σ(time)))
2	($C_{(0,0)} = 0$ AND f(p) AND NOT(σ(time))) OR ($C_{(0,0)} = 2$ AND ($C_{(1,0)} = 1$ OR $C_{(1,0)} = 2$)) OR ($C_{(0,0)} = 2$ AND σ(time))

Celdas de salida de la grilla

Tanto el borde derecho como el superior, contienen las celdas que permiten la salida de vehículos. Para cada uno de ellos se define una vecindad y comportamiento distinto.

Celdas del borde derecho

$\eta = 4$

N = { (0,0); (1,0); (-1,0); (0,-1) }

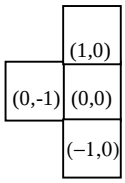


Figura 86 - Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)}$ = 1 AND σ(time)) OR ($C_{(0,0)}$ = 2 AND $C_{(1,0)}$ = 0 AND NOT(σ(time))) OR [$C_{(0,0)}$ = 0 AND ($C_{(-1,0)}$ = 0 OR ($C_{(-1,0)}$ = 2 AND σ(time))) AND ($C_{(0,-1)}$ = 0 OR ($C_{(0,-1)}$ = 1 AND NOT(σ(time))))]
1	($C_{(0,0)}$ = 0 AND $C_{(0,-1)}$ = 1 AND σ(time)) OR ($C_{(0,0)}$ = 1 AND NOT(σ(time)))
2	($C_{(0,0)}$ = 0 AND $C_{(-1,0)}$ = 2 AND NOT(σ(time))) OR ($C_{(0,0)}$ = 2 AND ($C_{(1,0)}$ = 1 OR $C_{(1,0)}$ = 2)) OR ($C_{(0,0)}$ = 2 AND σ(time))

Celdas del borde superior

η = 4

N = { (0,0); (0,1); (-1,0); (0,-1) }

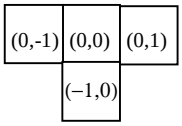


Figura 87 - Vecindario de la celda origen

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(0,0)}$ = 1 AND $C_{(0,1)}$ = 0 AND σ(time)) OR ($C_{(0,0)}$ = 2 AND NOT(σ(time))) OR [$C_{(0,0)}$ = 0 AND ($C_{(-1,0)}$ = 0 OR ($C_{(-1,0)}$ = 2 AND σ(time))) AND ($C_{(0,-1)}$ = 0 OR ($C_{(0,-1)}$ = 1 AND NOT(σ(time))))]
1	($C_{(0,0)}$ = 0 AND $C_{(0,-1)}$ = 1 AND σ(time)) OR ($C_{(0,0)}$ = 1 AND (NOT(σ(time)) OR $C_{(0,1)}$ > 0)
2	($C_{(0,0)}$ = 0 AND $C_{(-1,0)}$ = 2 AND NOT(σ(time))) OR ($C_{(0,0)}$ = 2 AND ($C_{(0,1)}$ = 1 OR $C_{(0,1)}$ = 2)) OR ($C_{(0,0)}$ = 2 AND σ(time))

B. Modelos de tráfico de dos carriles

En esta sección se presentan algunos modelos planteados para el tráfico de 2 carriles, los cuales pueden separarse según tengan igual o distinta dirección de circulación de autos.

B.1. Modelos con una única dirección

Cuando se modelan varios carriles con única dirección se deben establecer las reglas que determinan cuándo un auto cambia de carril. Las reglas resultantes pueden ser simétricas (sección B.1.1) o asimétricas (sección B.1.2), de acuerdo a la situación particular que se desee modelar.

B.1.1. Modelos con reglas simétricas

Como estos modelos presentan dos carriles paralelos que un vehículo puede utilizar, se deben definir reglas para pasar de uno al otro. Los modelos simétricos tienen la particularidad de utilizar las mismas reglas para ese pasaje. Este modelo tiene sentido en autopistas, donde los conductores no utilizan el carril que está hacia la derecha por defecto, pues en él circulan los autos lentos que ingresan a la autopista a través de rampas. De esta forma cuando encuentran un auto lento en el carril derecho, pasan al izquierdo y permanecen en él hasta que se acercan al auto de adelante o quieren salir de la autopista.

B.1.1.1. Modelos sin cruce de carriles

En esta categoría se agrupan los modelos con reglas simétricas sobre dos carriles unidireccionales, que no modelan intersecciones de carriles.

B.1.1.1.i. Modelos simples

En [RNSL96] se plantea un modelo con estas características mediante un autómata celular con los bordes de inicio y fin de cada carril conectados (periodic boundary conditions). Se realiza una extensión del modelo de [NS92] con vehículos de distintas velocidades máximas, planteado en la sección A.1.iii para poder representar las nuevas características. Se utilizan las 4 reglas definidas para un sólo carril en [NS92] y se agregan otras para moverse de un carril a otro. Así, la actualización del estado de un auto se realiza en 3 pasos:

1. El vehículo sólo se mueve hacia el costado, no avanza (usa las reglas que se definen a continuación).
2. Las 3 primeras reglas definidas para un sólo carril son chequeadas sobre cada uno de los carriles usando la configuración obtenida en el paso1.
3. Se actualiza la posición de los vehículos (regla 4 del modelo de un sólo carril).

Notación 9:

De aquí en adelante se notará como $\mathbf{gap_o}$ a la cantidad de posiciones vacías entre el vehículo y el auto de adelante en el otro carril.
$\mathbf{gap_{o,back}}$ notará a la cantidad de posiciones vacías entre el vehículo y el auto de atrás en el otro carril.

Las condiciones que deben ser satisfechas para que el auto cambie de carril son:

(T1) *gap* < l. Se fija si existe un auto adelante.

(T2) $gap_o > l_o$. Se fija si en el otro carril podrá acelerar más que en el actual.

(T3) $gap_{o,back} > l_{o,back}$. Se fija si en el otro carril estorbará el camino de otro auto que viene por atrás.

(T4) rand < p_{change}, con probabilidad p_{change} un auto cambia de carril.

Donde, l, l_o y $l_{o,back}$ representan a la longitud que indica hasta dónde ve el auto hacia adelante en su carril, hacia adelante y hacia atrás en el otro carril, respectivamente

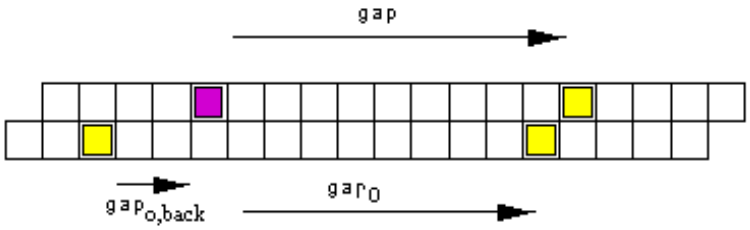


Figura 88 – Modelo

La inclusión de la regla (T4) se debe a que si todos los autos cambian de carril bajo las mismas circunstancias, y se ha formado una caravana sobre un carril, al evaluar el otro decidirán pasarse. De esta forma pasan de un carril a otro todos juntos y la caravana nunca se disuelve y tampoco logran adelantarse, esto se conoce como *efecto Ping Pong*. Por este motivo, se agrega una variable estocástica que permita el cambio de carril con cierta probabilidad, con lo cual sólo algunos podrán hacerlo.

Los autos se mantienen en sus carriles mientras no vean a nadie por adelante. Si detectan uno, chequean en el otro carril para pasarse en caso de ser posible. Luego del cambio, permanecen en ese carril hasta que encuentren otro obstáculo.

Este modelo se representa en forma similar al de [NS92] con distintas velocidades máximas, pero difiere en que es necesario realizar un paso previo que consiste en el cambio de carril para algunos vehículos. Aquí:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

S = { (v_{actual}, v_{max}) / $v_{actual} \in \{-1, 0, 1, 2, ..., v_{max}\}$, $v_{max} \in \{1, 2, ...,Max\}$ }

Donde,

v_{actual} = -1 representa la celda vacía,

v_{actual} = 0 representa que la celda tiene un vehículo con velocidad 0,

v_{actual} = 1 representa que la celda tiene un vehículo con velocidad 1, etc.

Max es una constante que se usa como parámetro de la simulación para indicar la máxima velocidad máxima posible.

n = 2

η = $l + Max + 1 + l_o + l_{o,back} + 1$

Donde,

l, l_o y $l_{o,back}$ es la longitud que indica hasta dónde ve el auto hacia adelante en su carril, hacia adelante y hacia atrás en el otro carril, respectivamente.

N = VecinosHaciaAdelante U VecinosHaciaAtrás U VecinosHaciaAdelanteOtroCarril U VecinosHaciaAtrásOtroCarril U { (0,0); (1,0) }

Donde,

VecinosHaciaAdelante ={(0,1); (0,2); (0,3);...; (0,l)}

VecinosHaciaAtrás ={(0,-1); (0,-2); (0,-3);...; (0,-Max)}

VecinosHaciaAdelanteOtroCarril ={(1,1); (1,2); (1,3);...; (1,l_o)}

VecinosHaciaAtrásOtroCarril ={(1,-1); (1,-2); (1,-3);...; (1,- $l_{o,back}$)}

(1,-$l_{o,back}$)		...	(1,-1)	(1,0)	(1,1)	...	(1,l_o)		
(0,-Max)		...	(0,-1)	(0,0)	(0,1)	...	(0,l)		

Figura 89 - Vecindario de la celda origen

Convención 6

Para los modelos de dos carriles se ha adoptado la siguiente convención, salvo que explícitamente se indique lo contrario. Dada una celda sobre alguno de los dos carriles, sus vecinos sobre su propio carril tienen como primera coordenada un 0 y los del otro carril un 1. También siguiendo la misma idea, la vecina (1,0), representa siempre la celda de al lado sobre el otro carril. Esta notación permite utilizar el mismo vecindario y las mismas reglas para las celdas de ambos carriles.

La función τ se aplica en tres pasos, en el primero se efectúan los cambios de carril (sin avanzar el vehículo), en el segundo se actualiza la velocidad de cada vehículo y en el tercero se actualiza la posición de acuerdo a esa nueva velocidad. Las reglas para estos dos últimos pasos, son idénticas a las definidas para un sólo carril del modelo [NS92] con distintas velocidades máximas y ya fueron definidas en la sección A.1.iii. No es necesario que sean modificadas pues son aplicadas sobre cada carril en forma totalmente independiente, una vez que se realizaron los cambios de carril correspondientes. Por tal motivo sólo se presentan a continuación las reglas nuevas, es decir, aquellas correspondientes al primer paso.

Notación 10:

De aquí en adelante se notará como **gap(i,j)**, a la cantidad de posiciones vacías entre la celda (i,j) y su primer vecina hacia adelante ocupada por un vehículo, es decir, aquella cuya v_{actual} sea mayor que –1.

gap_o(i,j), indicará la cantidad de posiciones vacías entre la celda (i,j) y su primer vecina hacia adelante ocupada por un vehículo sobre el otro carril.

$gap_{o,back}$(i,j), indicará la cantidad de posiciones vacías entre la celda (i,j) y su primer vecina hacia atrás ocupada por un vehículo sobre el otro carril.

change, representará a una función booleana que devuelve verdadero con probabilidad p_{change} (probabilidad de cambiar de carril), y en ese caso el auto puede cambiar de carril siempre y cuando se verifiquen las demás condiciones .

Reglas para cambio de carril (paso 1):

Nuevo Estado	Estado del vecindario
(-1,0)	$C_{(1,0)}.v_{actual}$ = -1 AND $C_{(0,0)}.v_{actual}$ >-1 AND change AND gap(0,0) < l AND $gap_o(0,0) > l_o$ AND $gap_{o,back}(0,0) > l_{o,back}$
$C_{(1,0)}$	$C_{(1,0)}.v_{actual}$ > -1 AND $C_{(0,0)}.v_{actual}$ = -1 AND change AND gap(1,0) < l AND $gap_o(1,0) > l_o$ AND $gap_{o,back}(1,0) > l_{o,back}$
$C_{(0,0)}$	[$C_{(0,0)}.v_{actual}$ >-1 AND NOT($C_{(1,0)}.v_{actual}$ = -1 AND change AND

	$gap(0,0) < l$ AND $gap_o(0,0) > l_o$ AND $gap_{o,back}(0,0) > l_{o,back}$)] OR [$C_{(0,0)}.v_{actual} = -1$ AND NOT($C_{(1,0)}.v_{actual} > -1$ AND change AND $gap(1,0) < l$ AND $gap_o(1,0) > l_o$ AND $gap_{o,back}(1,0) > l_{o,back}$)]

B.1.2. **Modelos con reglas asimétricas**

Los modelos asimétricos, a diferencia de los simétricos, no utilizan las mismas reglas para pasar de un carril a otro. En general, un carril se usa para adelantarse (pasar a un auto) y una vez realizado se vuelve al carril original, es decir, un carril (el derecho) se usa por defecto. En algunos países existen además otras restricciones que condicionan a las reglas de cambio de carril asimétricas; tal es el caso de Alemania, donde sólo se puede pasar a otro auto estando en el carril izquierdo, y el de Estados Unidos, donde no se prohibe pasar a otro auto estando en el carril derecho.

B.1.2.1. **Modelos sin cruce de carriles**

Aquí se ubican los modelos con reglas asimétricas sobre 2 carriles unidireccionales sin intersecciones.

B.1.2.1.i. **Modelos simples**

En [RNSL96] se presenta un modelo con reglas asimétricas para dos carriles unidireccionales. Éste consiste en una pequeña modificación al modelo simétrico propuesto en el mismo artículo que se ha descripto en la sección B.1.1.1 de este trabajo, cuyas reglas para cambiar carril son:

(T1) $gap < l$. Se fija si existe un auto adelante.
(T2) $gap_o > l_o$. Se fija si en el otro carril podrá acelerar más que en el actual.
(T3) $gap_{o,back} > l_{o,back}$. Se fija si en el otro carril estorbará el camino de otro auto que viene por atrás.
(T4) rand < p_{change}, con probabilidad p_{change} un auto cambia de carril.

Para construir la versión asimétrica se deben diferenciar los cambios de carril hacia la derecha de los que se dirigen hacia el carril izquierdo. Las condiciones que se deben verificar para los primeros son T2, T3 y T4, mientras que los que se realizan hacia el carril izquierdo deben verificar las cuatro condiciones. Esta modificación permite que un auto pase al carril izquierdo cuando desea pasar a un auto que le estorba el camino, pero una vez realizado vuelve al carril derecho, si tiene suficiente lugar. Es decir, no se queda en el carril izquierdo hasta que tenga adelante un auto más lento como sucede en la versión simétrica.

La representación de este modelo es similar a la del modelo simétrico y se define como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

$\mathbf{S} = \{ (v_{actual}, v_{max}) / v_{actual} \in \{-1, 0, 1, 2, ..., v_{max} \}, v_{max} \in \{1, 2, ...,Max \} \}$

Donde,

$v_{actual} = -1$ representa la celda vacía,
$v_{actual} = 0$ representa que la celda tiene un vehículo con velocidad 0,
$v_{actual} = 1$ representa que la celda tiene un vehículo con velocidad 1, etc.
Max es una constante que se usa como parámetro de la simulación para indicar la máxima velocidad máxima posible.

$\mathbf{n} = 2$

$\eta = l + Max + 1 + l_o + l_{o,back} + 1$

N = VecinosHaciaAdelante U VecinosHaciaAtrás U VecinosHaciaAdelanteOtroCarril U VecinosHaciaAtrásOtroCarril U { (0,0); (1,0) }

Donde,

VecinosHaciaAdelante = {(0,1); (0,2); (0,3);...; (0,l)}
VecinosHaciaAtrás = {(0,-1); (0,-2); (0,-3);...; (0,-Max)}
VecinosHaciaAdelanteOtroCarril = {(1,1); (1,2); (1,3);...; (1,l_o)}
VecinosHaciaAtrásOtroCarril = {(1,-1); (1,-2); (1,-3);...; (1,- $l_{o,back}$)}

(1,-$l_{o,back}$)	...	(1,-1)	(1,0)	(1,1)	...	(1,l_o)		
(0,-Max)	...	(0,-1)	(0,0)	(0,1)	...	(0,l)		

Figura 90 -Vecindario de la celda origen

Para la función **τ**, lo único que cambia respecto al modelo simétrico, es que las celdas del carril izquierdo tienen un comportamiento distinto a las del derecho, sólo para el cambio de carril. Las reglas de actualización de la velocidad y las de actualización de la posición siguen exactamente igual. Por tal motivo sólo se presentan a continuación las reglas nuevas, es decir, aquellas correspondientes al cambio de carril de las celdas del carril izquierdo y las correspondientes al cambio de carril de las celdas del carril derecho.

Reglas para cambio de carril para celdas del carril derecho:

Nuevo Estado	Estado del vecindario
(-1,0)	$C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} > -1$ AND change AND $Gap(0,0) < l$ AND $gap_o(0,0) > l_o$ AND $gap_{o,back}(0,0) > l_{o,back}$
$C_{(1,0)}$	$C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,0)}.v_{actual} = -1$ AND change AND $Gap(1,0) < l$ AND $gap_o(1,0) > l_o$ AND $gap_{o,back}(1,0) > l_{o,back}$
$C_{(0,0)}$	[$C_{(0,0)}.v_{actual} > -1$ AND NOT($C_{(1,0)}.v_{actual} = -1$ AND change AND $gap(0,0) < l$ AND $gap_o(0,0) > l_o$ AND $gap_{o,back}(0,0) > l_{o,back}$)] OR [$C_{(0,0)}.v_{actual} = -1$ AND NOT($C_{(1,0)}.v_{actual} > -1$ AND change AND $gap(1,0) < l$ AND $gap_o(1,0) > l_o$ AND $gap_{o,back}(1,0) > l_{o,back}$)]

Reglas para cambio de carril para celdas del carril izquierdo:

Nuevo Estado	Estado del vecindario
(-1,0)	$C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} > -1$ AND change AND $Gap_o(0,0) > l_o$ AND $gap_{o,back}(0,0) > l_{o,back}$
$C_{(1,0)}$	$C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,0)}.v_{actual} = -1$ AND change AND $Gap_o(1,0) > l_o$ AND $gap_{o,back}(1,0) > l_{o,back}$
$C_{(0,0)}$	[$C_{(0,0)}.v_{actual} > -1$ AND NOT($C_{(1,0)}.v_{actual} = -1$ AND change AND $gap_o(0,0) > l_o$ AND $gap_{o,back}(0,0) > l_{o,back}$)] OR [$C_{(0,0)}.v_{actual} = -1$ AND NOT($C_{(1,0)}.v_{actual} > -1$ AND change AND $gap_o(1,0) > l_o$ AND $gap_{o,back}(1,0) > l_{o,back}$)]

En [W95] se modelan reglas asimétricas, donde se prohibe que un auto pase a otro estando en el carril de la derecha. Para realizarlo extienden el modelo de [NS92], descripto en la sección A.1, con reglas que definen cuando se puede realizar el cambio de carril. De esta forma la actualización de estado se realiza en dos pasos:

1. *Decisión de cambio de carril.*
2. *Actualización según el modelo de un carril.*

Para tomar la decisión del cambio de carril se deben evaluar las siguientes condiciones en forma secuencial:

(4) $v_{o,back} < gap_{o,back}$, se requiere que el auto que viene por detrás en el otro carril, esté suficientemente lejos.

- Para cambiar *de derecha a izquierda* se necesitan verificar dos condiciones
  - (6) $v_{max} > gap$, se requiere que pueda acelerar lo suficiente para pasar al auto de adelante.
  - (7) $gap_o \geq gap$, se requiere que en el otro carril la situación no sea peor (o sea, si hay un auto hacia adelante que esté más lejos).
- Para cambiar *de izquierda a derecha* debe haber suficiente espacio en el carril derecho *e* izquierdo
  - (8) $v_{max} < gap - v_{off}$
  - (9) $v_{max} < gap_o - v_{off}$

Donde v_{off} se usa para cambiar el flujo o la densidad cuando el carril para adelantarse tiene más tráfico que el otro. Si aumenta el valor de v_{off}, entonces disminuye el flujo.

Este modelo se puede definir como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

$\mathbf{S} = \{ (v_{actual}, v_{max}) / v_{actual} \in \{-1, 0, 1, 2, ..., v_{max} \}, v_{max} \in \{1, 2, ...,Max \} \}$

Donde,

$v_{actual} = -1$ representa la celda vacía,

$v_{actual} = 0$ representa que la celda tiene un vehículo con velocidad 0,

$v_{actual} = 1$ representa que la celda tiene un vehículo con velocidad 1, etc.

Max es una constante que se usa como parámetro de la simulación para indicar la máxima velocidad máxima posible.

$\mathbf{n} = 2$

$\boldsymbol{\eta} = 4* Max + 2$

N = VecinosHaciaAdelante U VecinosHaciaAtrás U VecinosHaciaAdelanteOtroCarril U VecinosHaciaAtrásOtroCarril U { (0,0); (1,0) }

Donde,

VecinosHaciaAdelante = {(0,1); (0,2); (0,3);...; (0, Max)}

VecinosHaciaAtrás = {(0,-1); (0,-2); (0,-3);...; (0,-Max)}

VecinosHaciaAdelanteOtroCarril = {(1,1); (1,2); (1,3);...; (1, Max)}

VecinosHaciaAtrásOtroCarril = {(1,-1); (1,-2); (1,-3);...; (1, Max)}

(1, Max)	...	(1,-1)	(1,0)	(1,1)	...	(1, Max)		
(0,-Max)	...	(0,-1)	(0,0)	(0,1)	...	(0, Max)		

Figura 91 - Vecindario de la celda origen

La función τ se calcula en tres pasos, en el primero se efectúan los cambios de carril (sin avanzar el vehículo), en el segundo se actualiza la velocidad de cada vehículo y en el tercero la posición de acuerdo a esa nueva velocidad. Las reglas para estos dos últimos pasos, son idénticas a las definidas para un sólo carril del modelo [NS92] con distintas velocidades máximas y ya fueron definidas en la sección A.1.iii. No es necesario que sean modificadas pues son aplicadas sobre cada carril en forma totalmente independiente, una vez que se realizaron los cambios de carril correspondientes. Por tal motivo sólo se presentan a continuación las reglas nuevas, es decir, aquellas correspondientes al primer paso para las celdas del carril izquierdo y para las celdas del carril derecho.

Reglas para cambio de carril para celdas del carril derecho:

Nuevo Estado	Estado del vecindario
(-1,0)	$C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} > -1$ AND $C_{(1,gapo,back(0,0)-1)}.v_{actual} < gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{max} > gap(0,0)$ AND $gap_o(0,0) \geq gap(0,0)$
$C_{(1,0)}$	$C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,0)}.v_{actual} = -1$ AND $C_{(0,gapo,back(1,0)-1)}.v_{actual} < gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{max} < gap(1,0) - v_{off}$ AND $C_{(1,0)}.v_{max} < gap_o(1,0) - v_{off}$
$C_{(0,0)}$	$[C_{(0,0)}.v_{actual} > -1$ AND NOT($C_{(1,0)}.v_{actual} = -1$ AND $C_{(1,gapo,back(0,0)-1)}.v_{actual} < gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{max} > gap(0,0)$ AND $gap_o(0,0) \geq gap(0,0)$)] OR $[C_{(0,0)}.v_{actual} = -1$ AND NOT($C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,gapo,back(1,0)-1)}.v_{actual} < gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{max} < gap(1,0) - v_{off}$ AND $C_{(1,0)}.v_{max} < gap_o(1,0) - v_{off}$)]

Reglas para cambio de carril para celdas del carril izquierdo:

Nuevo Estado	Estado del vecindario
(-1,0)	$C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} > -1$ AND $C_{(1,gapo,back(0,0)-1)}.v_{actual} < gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{max} < gap(0,0) - v_{off}$ AND $C_{(0,0)}.v_{max} < gap_o(0,0) - v_{off}$
$C_{(1,0)}$	$C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,0)}.v_{actual} = -1$ AND $C_{(0,gapo,back(1,0)-1)}.v_{actual} < gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{max} > gap(1,0)$ AND $gap_o(1,0) \geq gap(1,0)$
$C_{(0,0)}$	$[C_{(0,0)}.v_{actual} > -1$ AND NOT($C_{(1,0)}.v_{actual} = -1$ AND $C_{(1,gapo,back(0,0)-1)}.v_{actual} < gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{max} < gap(0,0) - v_{off}$ AND $C_{(0,0)}.v_{max} < gap_o(0,0) - v_{off}$)] OR $[C_{(0,0)}.v_{actual} = -1$ AND NOT($C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,gapo,back(1,0)-1)}.v_{actual} < gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{max} > gap(1,0)$ AND $gap_o(1,0) \geq gap(1,0)$)]

Este modelo es analizado en [WNW97] y se proponen algunas modificaciones sobre las reglas para lograr un comportamiento más realista de la simulación. Se trata de evitar el amontonamiento de tráfico en el carril de la izquierda cuando la densidad de autos es grande.

Este efecto se produce con las condiciones anteriores (5 a 9) porque la regla (8) no puede ser satisfecha, entonces no funciona la vuelta del auto al carril derecho. Para resolver este problema se proponen las siguientes reglas:

(a) Las 4 reglas del modelo de un carril
(b) Las 5 reglas del modelo de dos carriles asimétrico de [W95]
(c) De las reglas para cambio de carril, (5) a (9): mantener las reglas (5), (6) y (7), eliminar la regla (8), modificar la (9) de la siguiente manera:
    (10) $v_{o,back,max} \leq gap_{o,back}$
    (11) $v \leq gap_o$ (espacio en la línea derecha)

Las reglas (c) se utilizan con probabilidad p_{l2r}, que es un número chico(por ejemplo, 0.02) y para el resto de los casos se usan las de (b).

Otra modificación que hacen al modelo es que por debajo de la velocidad v_{ban} (por ejemplo, 3) se puede pasar a un auto por el carril de la derecha con probabilidad $p_{brake}(1- p_{brake})$.

Este modelo se puede definir como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

$\mathbf{S} = \{ (v_{actual}, v_{max}) / v_{actual} \in \{-1, 0, 1, 2, ..., v_{max} \}, v_{max} \in \{1, 2, ...,Max \} \}$

Donde,

$v_{actual} = -1$ representa la celda vacía,
$v_{actual} = 0$ representa que la celda tiene un vehículo con velocidad 0,
$v_{actual} = 1$ representa que la celda tiene un vehículo con velocidad 1, etc.
Max es una constante que se usa como parámetro de la simulación para indicar la máxima velocidad máxima posible.

$\mathbf{n} = 2$

$\boldsymbol{\eta} = 4* Max + 2$

N = VecinosHaciaAdelante U VecinosHaciaAtrás U VecinosHaciaAdelanteOtroCarril U VecinosHaciaAtrásOtroCarril U { (0,0); (1,0) }

Donde,

VecinosHaciaAdelante = {(0,1); (0,2); (0,3);...; (0, Max)}
VecinosHaciaAtrás = {(0,-1); (0,-2); (0,-3);...; (0,-Max)}
VecinosHaciaAdelanteOtroCarril = {(1,1); (1,2); (1,3);...; (1, Max)}
VecinosHaciaAtrásOtroCarril = {(1,-1); (1,-2); (1,-3);...; (1, Max)}

(1, Max)	...	(1,-1)	(1,0)	(1,1)	...	(1, Max)		
(0,-Max)	...	(0,-1)	(0,0)	(0,1)	...	(0, Max)		

Figura 92 - Vecindario de la celda origen

La función $\boldsymbol{\tau}$ se calcula en tres pasos, en el primero se efectúan los cambios de carril (sin avanzar el vehículo), en el segundo se actualiza la velocidad de cada vehículo y en el tercero la posición de acuerdo a esa nueva velocidad. Las reglas para estos dos últimos pasos, son idénticas a las definidas para un sólo carril del modelo [NS92] con distintas velocidades máximas y ya fueron definidas en la sección A.1.iii. No es necesario que sean modificadas pues son aplicadas sobre cada carril en forma totalmente independiente, una vez que se realizaron los cambios de carril correspondientes. Por tal motivo sólo se presentan a continuación las reglas

nuevas, es decir, aquellas correspondientes al primer paso para las celdas del carril izquierdo y para las celdas del carril derecho.

Reglas para cambio de carril para celdas del carril derecho:

Nuevo Estado	Estado del vecindario
(-1,0)	$C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} > -1$ AND $C_{(1,gapo,back(0,0)-1)}.v_{actual} < gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{max} > gap(0,0)$ AND $gap_o(0,0) \geq gap(0,0)$
$C_{(1,0)}$	[NOT (p_{l2r}) AND $C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,0)}.v_{actual} = -1$ AND $C_{(0,gapo,back(1,0)-1)}.v_{actual} < gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{max} < gap(1,0) - v_{off}$ AND $C_{(1,0)}.v_{max} < gap_o(1,0) - v_{off}$] OR [p_{l2r} AND $C_{(0,0)}.v_{actual} = -1$ AND $C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,gapo,back(1,0)-1)}.v_{max} \leq gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$]
$C_{(0,0)}$	[$C_{(0,0)}.v_{actual} > -1$ AND NOT($C_{(1,0)}.v_{actual} = -1$ AND $C_{(1,gapo,back(0,0)-1)}.v_{actual} < gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{max} > gap(0,0)$ AND $gap_o(0,0) \geq gap(0,0)$)] OR [NOT (p_{l2r}) AND $C_{(0,0)}.v_{actual} = -1$ AND NOT($C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,gapo,back(1,0)-1)}.v_{actual} < gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{max} < gap(1,0) - v_{off}$ AND $C_{(1,0)}.v_{max} < gap_o(1,0) - v_{off}$)] OR [p_{l2r} AND $C_{(0,0)}.v_{actual} = -1$ AND NOT($C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,gapo,back(1,0)-1)}.v_{max} \leq gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$)]

Donde,

v_{off}, es parámetro del modelo.

p_{l2r}, es verdadero con cierta probabilidad y significa que se deben usar el nuevo conjunto de reglas para cambiar del carril izquierdo al derecho.

Reglas para cambio de carril para celdas del carril izquierdo:

Nuevo Estado	Estado del vecindario
(-1,0)	[NOT (p_{l2r}) AND $C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} > -1$ AND $C_{(1,gapo,back(0,0)-1)}.v_{actual} < gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{max} < gap(0,0) - v_{off}$ AND $C_{(0,0)}.v_{max} < gap_o(0,0) - v_{off}$] OR [p_{l2r} AND $C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} > -1$ AND $C_{(1,gapo,back(0,0)-1)}.v_{max} \leq gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$]
$C_{(1,0)}$	$C_{(1,0)}.v_{actual} > -1$ AND $C_{(0,0)}.v_{actual} = -1$ AND $C_{(0,gapo,back(1,0)-1)}.v_{actual} < gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{max} > gap(1,0)$ AND $gap_o(1,0) \geq gap(1,0)$
$C_{(0,0)}$	[NOT (p_{l2r}) AND $C_{(0,0)}.v_{actual} > -1$ AND NOT($C_{(1,0)}.v_{actual} = -1$ AND $C_{(1,gapo,back(0,0)-1)}.v_{actual} < gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{max} < gap(0,0) - v_{off}$ AND $C_{(0,0)}.v_{max} < gap_o(0,0) - v_{off}$)] OR [p_{l2r} AND $C_{(0,0)}.v_{actual} > -1$ AND NOT($C_{(1,0)}.v_{actual} = -1$ AND

	$C_{(1,gapo,back(0,0)-1)}.v_{max} \leq gap_{o,back}(0,0)$ AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$)] OR [$C_{(0,0)}.v_{actual}$ = -1 AND NOT($C_{(1,0)}.v_{actual}$ > -1 AND $C_{(0,gapo,back(1,0)-1)}.v_{actual} < gap_{o,back}(1,0)$ AND $C_{(1,0)}.v_{max} > gap(1,0)$ AND $gap_o(1,0) \geq gap(1,0)$)]

B.1.2.1.ii. **Modelos con vehículos especiales**

En [NWWS97] se propone un modelo para el tráfico sobre dos carriles con dos conjuntos distintos de reglas asimétricas. Este modelo en vez de basarse en el gap de separación entre los vehículos utiliza sus velocidades para las reglas de comportamiento.

Para modelar el comportamiento del tráfico se definen reglas que tengan en cuenta dos aspectos, un *criterio de seguridad* y la *motivación* para cambiar de carril. El criterio de seguridad significa dejar suficiente espacio entre todos los vehículos, al realizar el cambio de carril. De esta forma se requiere que haya un lugar de $gap_o + gap_{o,back} + 1$ en el carril al que se desea pasar. Este criterio se representa con un intervalo de la siguiente forma: $[-gap_{o,back}, gap_o]$ y para este modelo se ha utilizado $gap_o = v$ y $gap_{o,back} = v_{max}$. Por otro lado, están las reglas que motivan a que un vehículo cambie de carril, que son definidas para dos modelos distintos, en el primero se prohibe pasar a otro auto estando en el carril derecho y en el segundo se permite dicha acción. Las reglas para el primer modelo, que rige en autopistas alemanas, son las siguientes:

$$v_r \leq v \text{ OR } v_l \leq v \qquad (1)$$
$$v_r > v \text{ AND } v_l > v \qquad (2)$$

Donde, v_r y v_l son las velocidades de los autos más próximos dentro de una distancia, d, en el carril derecho e izquierdo, respectivamente. Si no hay ningún auto dentro de esa distancia, v_r y v_l toman el valor infinito. La regla (1) corresponde al cambio de carril *hacia la izquierda*, y surge de no permitir pasar a otro auto estando en el carril derecho, si hay un auto lento en el carril izquierdo para evitar pasarlo hay que colocarse detrás de él. De esta forma el cambio de carril se da si hay un auto lento en ese carril o en el propio. La regla (2) corresponde al cambio de carril *hacia la derecha*, y surge de hacer la negación lógica de las condiciones que permiten el cambio inverso. Pues la razón del cambio inverso ya no existe y se vuelve al carril original (derecho). Esto significa que se vuelve al carril derecho tan pronto como las velocidades de los autos de adelante son suficientemente altas.

Las reglas para el segundo modelo, que rige en autopistas de Estados Unidos, son las siguientes:

$$v_r \leq v \text{ AND } v_r \leq v_l \qquad (3)$$
$$v_r > v \text{ OR } v_r > v_l \qquad (4)$$

La regla (3) corresponde al cambio de carril *hacia la izquierda*, en este caso el carril izquierdo sólo es más atractivo si el tráfico en él es más rápido que en el propio. De esta forma el cambio de carril *hacia la izquierda* se da si hay un auto lento en el carril propio y además si en otro carril el auto de adelante va más rápido. La regla (4) corresponde al cambio de carril *hacia la derecha*, y surge de negar la regla (3). Esto significa que se vuelve al carril derecho si hay un auto más rápido que uno mismo en el carril derecho, o si el tráfico en el carril derecho es más rápido que en el otro.

Como en modelos anteriores, las reglas planteadas sólo deciden si el vehículo cambiará de carril; por eso es necesario agregar aquellas que permitan los movimientos de avance. En definitiva, se debe considerar el criterio de seguridad, la motivación para el cambio de carril y por último el movimiento de avance; que se llevan a cabo en dos pasos:

1. Decisión de cambio de carril.
   En los *pasos de tiempo pares*, se realizan los cambios de carril de derecha a izquierda. Todos los vehículos que satisfacen el criterio de motivación (regla 1) y el criterio de seguridad ([-v_{max},v]), se mueven en forma simultánea a la izquierda.
   En los *pasos de tiempo impares*, se realizan los cambios de carril de izquierda a derecha. Todos los vehículos que satisfacen el criterio de motivación (regla 2) y el criterio de seguridad ([-v_{max},v]), se mueven en forma simultánea a la derecha.
   La separación en ciclos pares e impares se utiliza para permitir la extensión del modelo a 3 carriles; pues en esa situación se puede dar que un vehículo del carril izquierdo y otro del derecho, se muevan a la misma celda del carril del medio.
   La cantidad de posiciones, d, que se observan hacia adelante en las reglas (1) y (2), juegan un rol crítico. Si d es grande, se adquiere la tendencia de ir al carril izquierdo muchos antes de acercarse a un auto lento; lo que lleva a una fuerte inversión del carril en bajas densidades. Por lo tanto d puede ser usado para ajustar la inversión de densidad.
2. Reglas para el movimiento de avance.
   Son las mismas reglas descriptas en secciones anteriores para el movimiento en un sólo carril, que fueron introducidas en [NS92].

Estas reglas son extendidas para lograr un comportamiento más realista del modelo, agregando camiones, simetría de las reglas en altas densidades y una cierta demora. La *demora* en las reglas se introduce debido que a la inversión máxima del uso de carriles se alcanza en densidades demasiado bajas comparadas con los datos reales. Así se agrega una demora, Δ, para volver a la derecha y las nuevas reglas serían:

$$v_r > v + \Delta \text{ AND } v_l > v + \Delta \qquad (7)$$

$$v_r \leq v \text{ OR } v_l \leq v \qquad (8)$$

Otra modificación que se le agrega al modelo es que las reglas de cambio de carril sean *simétricas* en altas densidades/bajas velocidades. Esto surge porque el tráfico nunca vuelve a un uso equivalente de carriles, luego de la inversión. Así, un vehículo con v = 0, sólo chequea si la velocidad en el otro carril es mayor que en su propio carril y si es así intenta cambiar de carril (de acuerdo al criterio de seguridad). La última característica que se agrega al modelo es la presencia de *autos lentos o camione*, dándole al 10% de vehículos una velocidad máxima más baja.

Este modelo se puede representar como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

$\mathbf{S} = \{ (v_{actual}, v_{max}) / v_{actual} \in \{-1, 0, 1, 2, ..., v_{max} \}, v_{max} \in \{1, 2, ...,Max \} \}$

Donde,

v_{actual} = -1 representa la celda vacía,

v_{actual} = 0 representa que la celda tiene un vehículo con velocidad 0,

v_{actual} = 1 representa que la celda tiene un vehículo con velocidad 1, etc.

Max es una constante que se usa como parámetro de la simulación para indicar la máxima velocidad máxima posible.

$\mathbf{n} = 2$

$\boldsymbol{\eta} = 4*Max + 2$

N = VecinosHaciaAdelante U VecinosHaciaAtrás U VecinosHaciaAdelanteOtroCarril U VecinosHaciaAtrásOtroCarril U { (0,0); (1,0) }

Donde,

VecinosHaciaAdelante = {(0,1); (0,2); (0,3);...; (0, Max)}

VecinosHaciaAtrás = {(0,-1); (0,-2); (0,-3);...; (0,-Max)}

VecinosHaciaAdelanteOtroCarril = {(1,1); (1,2); (1,3);...; (1, Max)}

VecinosHaciaAtrásOtroCarril = {(1,-1); (1,-2); (1,-3);...; (1, Max)}

(1, Max)	...	(1,-1)	(1,0)	(1,1)	...	(1, Max)			
(0,-Max)	...	(0,-1)	(0,0)	(0,1)	...	(0, Max)			

Figura 93 - Vecindario de la celda origen

La función τ se calcula en tres pasos, en el primero se efectúan los cambios de carril (sin avanzar el vehículo), en el segundo se actualiza la velocidad de cada vehículo y en el tercero se actualiza la posición de acuerdo a esa nueva velocidad. Las reglas para estos dos últimos pasos, son idénticas a las definidas para un sólo carril del modelo [NS92] con distintas velocidades máximas y ya fueron definidas en la sección A.1.iii. No es necesario que sean modificadas pues son aplicadas sobre cada carril en forma totalmente independiente, una vez que se realizaron los cambios de carril correspondientes. Por tal motivo sólo se presentan a continuación las reglas nuevas, es decir, aquellas correspondientes al primer paso para las celdas del carril izquierdo y para las celdas del carril derecho.

Reglas para cambio de carril para celdas del carril derecho: reglas 1) y 2) con el criterio de seguridad

Nuevo Estado	Estado del vecindario
(-1,0)	$[C_{(0,0)}.v_{actual} = -1$ AND NOT($C_{(1,0)}.v_{actual} > -1$ AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$ AND $C_{(1,0)}.v_{max} \leq gap_{o,back}(1,0)$ AND $C_{(0,gapo(1,0)+1)}.v_{actual} > C_{(1,0)}.v_{actual}$ AND $C_{(1,gap(1,0)+1)}.v_{actual} > C_{(1,0)}.v_{actual}$)] OR [$C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} > -1$ AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$ AND $C_{(0,0)}.v_{max} \leq gap_{o,back}(0,0)$ AND ($C_{(0,gap(0,0)+1)}.v_{actual} \leq C_{(0,0)}.v_{actual}$ OR $C_{(1,gapo(0,0)+1)}.v_{actual} \leq C_{(0,0)}.v_{actual}$)]
$C_{(1,0)}$	$C_{(0,0)}.v_{actual} = -1$ AND $C_{(1,0)}.v_{actual} > -1$ AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$ AND $C_{(1,0)}.v_{max} \leq gap_{o,back}(1,0)$ AND $C_{(0,gapo(1,0)+1)}.v_{actual} > C_{(1,0)}.v_{actual}$ AND $C_{(1,gap(1,0)+1)}.v_{actual} > C_{(1,0)}.v_{actual}$
$C_{(0,0)}$	$C_{(0,0)}.v_{actual} > -1$ AND NOT($C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$ AND $C_{(0,0)}.v_{max} \leq gap_{o,back}(0,0)$ AND ($C_{(0,gap(0,0)+1)}.v_{actual} \leq C_{(0,0)}.v_{actual}$ OR $C_{(1,gapo(0,0)+1)}.v_{actual} \leq C_{(0,0)}.v_{actual}$))

Reglas para cambio de carril para celdas del carril izquierdo:

Nuevo Estado	Estado del vecindario
(-1,0)	$[C_{(0,0)}.v_{actual} = -1$ AND NOT($C_{(1,0)}.v_{actual} > -1$ AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$ AND $C_{(1,0)}.v_{max} \leq gap_{o,back}(1,0)$ AND ($C_{(1,gap(1,0)+1)}.v_{actual} \leq C_{(1,0)}.v_{actual}$ OR $C_{(0,gapo(1,0)+1)}.v_{actual} \leq C_{(1,0)}.v_{actual}$))]

	OR [$C_{(0,0)}.v_{actual}$ >-1 AND $C_{(1,0)}.v_{actual}$ = -1 AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$ AND $C_{(0,0)}.v_{max} \leq gap_{o,back}(0,0)$ AND $C_{(1,gapo(0,0)+1)}.v_{actual} > C_{(0,0)}.v_{actual}$ AND $C_{(0,gap(0,0)+1)}.v_{actual} > C_{(0,0)}.v_{actual}$]
$C_{(1,0)}$	$C_{(0,0)}.v_{actual}$ = -1 AND $C_{(1,0)}.v_{actual}$ > -1 AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$ AND $C_{(1,0)}.v_{max} \leq gap_{o,back}(1,0)$ AND ($C_{(1,gap(1,0)+1)}.v_{actual} \leq C_{(1,0)}.v_{actual}$ OR $C_{(0,gapo(1,0)+1)}.v_{actual} \leq C_{(1,0)}.v_{actual}$)
$C_{(0,0)}$	$C_{(0,0)}.v_{actual}$ >-1 AND NOT($C_{(1,0)}.v_{actual}$ = -1 AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$ AND $C_{(0,0)}.v_{max} \leq gap_{o,back}(0,0)$ AND $C_{(1,gapo(0,0)+1)}.v_{actual} > C_{(0,0)}.v_{actual}$ AND $C_{(0,gap(0,0)+1)}.v_{actual} > C_{(0,0)}.v_{actual}$)]

<u>Reglas para cambio de carril para celdas del carril derecho</u>: reglas 3) y 4) con el criterio de seguridad

Nuevo Estado	Estado del vecindario
(-1,0)	[$C_{(0,0)}.v_{actual}$ =-1 AND NOT($C_{(1,0)}.v_{actual}$ > -1 AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$ AND $C_{(1,0)}.v_{max} \leq gap_{o,back}(1,0)$ AND ($C_{(0,gapo(1,0)+1)}.v_{actual} > C_{(1,0)}.v_{actual}$ OR $C_{(0,gapo(1,0)+1)}.v_{actual} > C_{(1,\ gap(1,0)+1)}.v_{actual}$))] OR [$C_{(1,0)}.v_{actual}$ = -1 AND $C_{(0,0)}.v_{actual}$ >-1 AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$ AND $C_{(0,0)}.v_{max} \leq gap_{o,back}(0,0)$ AND ($C_{(0,gap(0,0)+1)}.v_{actual} \leq C_{(0,0)}.v_{actual}$ AND $C_{(0,gap(0,0)+1)}.v_{actual} \leq C_{(1,\ gapo(0,0)+1)}.v_{actual}$)]
$C_{(1,0)}$	$C_{(0,0)}.v_{actual}$ =-1 AND $C_{(1,0)}.v_{actual}$ > -1 AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$ AND $C_{(1,0)}.v_{max} \leq gap_{o,back}(1,0)$ AND ($C_{(0,gapo(1,0)+1)}.v_{actual} > C_{(1,0)}.v_{actual}$ OR $C_{(0,gapo(1,0)+1)}.v_{actual} > C_{(1,\ gap(1,0)+1)}.v_{actual}$)
$C_{(0,0)}$	$C_{(0,0)}.v_{actual}$ >-1 AND NOT($C_{(1,0)}.v_{actual}$ = -1 AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$ AND $C_{(0,0)}.v_{max} \leq gap_{o,back}(0,0)$ AND ($C_{(0,gap(0,0)+1)}.v_{actual} \leq C_{(0,0)}.v_{actual}$ AND $C_{(0,gap(0,0)+1)}.v_{actual} \leq C_{(1,\ gapo(0,0)+1)}.v_{actual}$))

<u>Reglas para cambio de carril para celdas del carril izquierdo</u>:

Nuevo Estado	Estado del vecindario
(-1,0)	[$C_{(0,0)}.v_{actual}$ = -1 AND NOT($C_{(1,0)}.v_{actual}$ >-1 AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$ AND $C_{(1,0)}.v_{max} \leq gap_{o,back}(1,0)$ AND $C_{(1,gap(1,0)+1)}.v_{actual} \leq C_{(1,0)}.v_{actual}$ AND $C_{(1,gap(1,0)+1)}.v_{actual} \leq C_{(0,\ gapo(1,0)+1)}.v_{actual}$)] OR [$C_{(0,0)}.v_{actual}$ >-1 AND $C_{(1,0)}.v_{actual}$ = -1 AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$ AND $C_{(0,0)}.v_{max} \leq gap_{o,back}(0,0)$ AND ($C_{(1,gapo(0,0)+1)}.v_{actual} > C_{(0,0)}.v_{actual}$ OR $C_{(1,gapo(0,0)+1)}.v_{actual} > C_{(0,\ gap(0,0)+1)}.v_{actual}$)]
$C_{(1,0)}$	$C_{(0,0)}.v_{actual}$ = -1 AND $C_{(1,0)}.v_{actual}$ >-1 AND $C_{(1,0)}.v_{actual} \leq gap_o(1,0)$ AND $C_{(1,0)}.v_{max} \leq gap_{o,back}(1,0)$ AND $C_{(1,gap(1,0)+1)}.v_{actual} \leq C_{(1,0)}.v_{actual}$ AND

	$C_{(1,gap(1,0)+1)}.v_{actual} \leq C_{(0,\ gapo(1,0)+1)}.v_{actual}$
$C_{(0,0)}$	$C_{(0,0)}.v_{actual} > -1$ AND NOT($C_{(1,0)}.v_{actual} = -1$ AND $C_{(0,0)}.v_{actual} \leq gap_o(0,0)$ AND $C_{(0,0)}.v_{max} \leq gap_{o,back}(0,0)$ AND ($C_{(1,gapo(0,0)+1)}.v_{actual} > C_{(0,0)}.v_{actual}$ OR $C_{(1,gapo(0,0)+1)}.v_{actual} > C_{(0,\ gap(0,0)+1)}.v_{actual}$))

B.2. **Modelos con dos direcciones**

En esta sección se presentan modelo de 2 carriles con distinta dirección que tienen en cuenta, por ejemplo, la utilización del carril de dirección contraria para pasar a un auto, y una vez que terminó debe volver a su mano. Éstos pueden permitir adelantamiento de autos en ambos carriles, adelantamiento en alguno de los 2 carriles o no permitirlo en ninguno. Estas tres alternativas pueden ser combinadas en distintas secciones del camino, por ejemplo, las curvas de una ruta se representan con carriles que no permiten pasar de uno a otro. Además para los modelos que definen cruces es interesante representar el giro sobre el tráfico de dirección contraria.

B.2.1. **Modelos sin cruce de carriles**

En estos modelos no se considera el comportamiento de los vehículos al llegar a las intersecciones, se representan carriles que no se cruzan.

B.2.1.i. **Modelos simples**

En [SG98] se modela el tráfico de dos carriles con dirección opuesta a través de un autómata celular de 2 filas y con bordes conectados entre sí (periodic boundary conditions). Cada celda tiene un estado v, tal que $v \in \{-(v_{max} + 1),...,v_{max} + 1\}$, donde:

- 0 representa la ausencia de un vehículo;
- +1 o -1 representa un auto detenido;
- +2,-2 representa un auto moviéndose con velocidad 1 en dirección positiva, y un auto moviéndose con velocidad 1 en dirección negativa, respectivamente;
- y así siguiendo.

Los autos y carriles son etiquetados con "+" y "-". Un auto "+" en un carril "+", está en su propio carril; mientras que auto "+" en un carril "-", está en el carril de adelantamiento. En forma simétrica se definen el carril propio y el de adelantamiento para los autos "-".

Para definir las reglas del comportamiento de los vehículos se utiliza los siguientes parámetros del modelo, l y l_o es la longitud que indica hasta dónde ve el auto hacia adelante en su carril, y hacia adelante en el otro carril, respectivamente. Otros parámetros son l_{pass}, $l_{security}$ y D_{limit}, que indican el espacio que debe haber sobre el carril para que el auto intente un adelantamiento, el espacio que debe haber sobre el carril de adelantamiento para que el auto no esté en peligro y la máxima densidad local para un cambio de carril seguro, respectivamente.

Para la descripción de las reglas del comportamiento de los vehículos, v_a y $v_{a,o}$ denotan la velocidad del auto de adelante en el mismo carril y en el opuesto, respectivamente. También se define signo, como una función que devuelve "+1" o "-1" según el rótulo de los autos; H, como una función que devuelve true sii el vehículo está sobre su propio carril; oncoming, como una función que devuelve true sii $signo(v_a) \neq signo(v)$; y por último, D_l (*densidad local*), como una función que devuelve la cantidad de posiciones hacia adelante de $l_{density} = (2v_{max} + l)$ que están ocupadas.

El movimiento de los vehículos se realiza en dos pasos:

1) Cambio de carril
2) Movimiento de avance

Reglas para el cambio de carril:

(1) Si (H AND (gap < l_{pass}) AND (gap_o > $l_{security}$) AND ($D_l \leq D_{limit}$) AND (rand < p_{change})) entonces cambiar de carril.

Esta regla rige sobre los autos que están en su propio carril. Si el auto de adelante está a una distancia menor a l_{pass}, el auto de atrás buscará adelantarse. Pero para hacerlo debe haber suficiente lugar en el otro carril, el número de autos en frente del vehículo que se quiere pasar debe ser pequeño; y se debe considerar la probabilidad de cambiar de carril (p_{change}).

(2) Si (NOT(H) AND ((gap < $l_{security}$) OR ($gap_o \geq l_{pass}$))) entonces cambiar de carril.

Esta regla concierne a los autos en el medio del adelantamiento. Ellos retornan a su propio carril si son forzados por el aproximamiento de un auto en sentido contrario, o si hay suficiente espacio en su propio carril para volver sin frenar.

Reglas para el movimiento de avance:

(3) Si ($|v| \neq v_{max}$) entonces v = v + signo(v)
Si se puede, se acelera el auto.

(4) Si (oncoming AND ($gap \leq (2\ v_{max} - 1)$)) entonces v = $\lfloor gap/2 \rfloor$
Frena rápidamente si hay un auto muy cerca en sentido opuesto.

(5) Si (NOT(oncoming) AND ($|v| > gap$)) entonces v = signo(v)* (gap)
Frena si se está acercando a otro auto en su carril de origen.

(6) Si (H AND |v| > 1 AND rand < p_{brake} AND NOT(oncoming)) entonces v=v-signo(v)
Frena en forma aleatoria, si el auto se encuentre en su propio carril, pero si se está adelantando nunca desacelera aleatoriamente.

(7) Si (H AND oncoming AND |v| > 1) entonces v = v - signo(v)
Rompe la simetría entre los carriles, y así previene la formación de "super embotellamientos".

Un vehículo tiene aceleración constante mientras se esté adelantando, para ello p_{brake} es seteado a 0 durante ese proceso. Si se detecta que un auto se acerca, el vehículo que se está adelantando, debe volver a su carril en forma inmediata. En la vida real, un auto no intentará adelantarse salvo que lo pueda lograr exitosamente. Para modelar este fenómeno, se permite que cada vehículo mida la *densidad local*, que es la densidad de autos en frente del vehículo que quiere adelantarse. Si la densidad local es suficientemente baja, el vehículo tiene una buena chance de completar el adelantamiento y se permite que lo intente.

Este modelo se puede represnetar como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

$\mathbf{S} = \{ -v_{max, \ldots}, -1, 0, 1, 2, \ldots, v_{max} \}$

Donde,

0 representa la celda vacía,

+1 o -1 representa la presencia de un auto detenido en la celda,

+2,-2 representa la presencia de un auto con velocidad 1 en dirección positiva, y un auto con velocidad 1 en dirección negativa , respectivamente, en la celda. La velocidad positiva indica que el auto está en su propio carril, si es negativa se encuentra en el carril de dirección contraria.

y así siguiendo.

n = 2

$\eta = l_o + l + v_{max} + 2$

N = VecinosHaciaAdelante U VecinosHaciaAtrás U VecinosHaciaAdelanteOtroCarril U { (0,0); (1,0) }

Donde,

VecinosHaciaAdelante = {(0,1); (0,2); (0,3);...; (0, l)}

VecinosHaciaAtrás = {(0,-1); (0,-2); (0,-3);...; (0,- v_{max})}

VecinosHaciaAdelanteOtroCarril = {(1,1); (1,2); (1,3);...; (1, l_o)}

				(1,0)	(1,1)	...	(1, l_o)		
	(0,-v_{max})	...	(0,-1)	(0,0)	(0,1)	...	(0, l)		

Figura 94 - Vecindario de la celda origen

La función τ se aplica en tres pasos, en el primero se efectúan los cambios de carril (sin avanzar el vehículo), en el segundo se actualiza la velocidad de cada vehículo y en el tercero la posición de acuerdo a la nueva velocidad.

Reglas para cambio de carril (paso 1)

Nuevo Estado	Estado del vecindario
0	[$C_{(0,0)} = 0$ AND NOT($C_{(1,0)} > 0$ AND $gap(1,0) < l_{pass}$ AND $gap_o(1,0) > l_{security}$ AND $D_l(1,0) \leq D_{limit}$ AND change)] OR [$C_{(0,0)} > 0$ AND $C_{(1,0)} = 0$ AND $gap(0,0) < l_{pass}$ AND $gap_o(0,0) > l_{security}$ AND $D_l(0,0) \leq D_{limit}$ AND change] OR [$C_{(0,0)} = 0$ AND NOT($C_{(1,0)} < 0$ AND ($gap(1,0) < l_{security}$ OR $gap_o(1,0) \geq l_{pass}$))] OR [$C_{(0,0)} < 0$ AND $C_{(1,0)} = 0$ AND ($gap(0,0) < l_{security}$ OR $gap_o(1,0) \geq l_{pass}$)]
$-C_{(1,0)}$	[$C_{(0,0)} = 0$ AND $C_{(1,0)} > 0$ AND $gap(1,0) < l_{pass}$ AND $gap_o(1,0) > l_{security}$ AND $D_l(1,0) \leq D_{limit}$ AND change] OR [$C_{(0,0)} = 0$ AND $C_{(1,0)} < 0$ AND ($gap(1,0) < l_{security}$ OR $gap_o(1,0) \geq l_{pass}$))]
$C_{(0,0)}$	[$C_{(0,0)} > 0$ AND NOT($C_{(1,0)} = 0$ AND $gap(0,0) < l_{pass}$ AND $gap_o(0,0) > l_{security}$ AND $D_l(0,0) \leq D_{limit}$ AND change)] OR [$C_{(0,0)} < 0$ AND NOT($C_{(1,0)} = 0$ AND ($gap(0,0) < l_{security}$ OR $gap_o(1,0) \geq l_{pass}$))]

Donde,

l_{pass}, $l_{security}$ y D_{limit} son parámetros del modelo que representan, la cantidad de posiciones vacías para hacer un cambio de carril exitoso y seguro.

$D_l(i,j)$ es la cantidad de posiciones ocupadas sobre el carril de (i,j) y hacia adelante, considerando $2*v_{max}+1$ posiciones. Es decir, aquellas cuyo estado sea distinto de cero.

Reglas actualizar la velocidad (paso 2)

Una vez efectuados los cambios de carril correspondientes, en base al estado obtenido en ese paso, se procede a la actualización de las velocidades.

Nuevo Estado	Estado del vecindario
$C_{(0,0)}$ +signo($C_{(0,0)}$)	[$\|C_{(0,0)}\| < v_{max}$ AND Oncoming AND NOT(gap ≤ 2*v_{max}-1) AND NOT($\|C_{(0,0)}$ + signo($C_{(0,0)}$)| > 1 AND $C_{(0,0)}$ + signo($C_{(0,0)}$) > 0)] OR [$\|C_{(0,0)}\| < v_{max}$ AND NOT(Oncoming) AND NOT($\|C_{(0,0)}$ + signo($C_{(0,0)}$)| > gap) AND NOT($C_{(0,0)}$ + signo($C_{(0,0)}$) > 0 AND |$C_{(0,0)}$ + signo($C_{(0,0)}$)	> 1 AND brake)]
$C_{(0,0)}$	[$\|C_{(0,0)}\| = v_{max}$ AND Oncoming AND NOT(gap ≤ 2*v_{max}-1) AND NOT($\|C_{(0,0)}\|$ > 1 AND $C_{(0,0)}$ > 0)] OR [$\|C_{(0,0)}\| < v_{max}$ AND Oncoming AND NOT(gap ≤ 2*v_{max}-1) AND |$C_{(0,0)}$ + signo($C_{(0,0)}$)| > 1 AND $C_{(0,0)}$ + signo($C_{(0,0)}$) > 0] OR [$\|C_{(0,0)}\| = v_{max}$ AND NOT(Oncoming) AND NOT($\|C_{(0,0)}\|$ > gap) AND NOT($C_{(0,0)}$ > 0 AND |$C_{(0,0)}$	> 1 AND brake)] OR [$\|C_{(0,0)}\| < v_{max}$ AND NOT(Oncoming) AND NOT(|$C_{(0,0)}$|+1 > gap) AND $C_{(0,0)}$+signo($C_{(0,0)}$) > 0 AND |$C_{(0,0)}$+signo($C_{(0,0)}$)	> 1 AND brake]
⌊gap/2⌋	[Oncoming AND gap ≤ 2*v_{max}-1 AND NOT(⌊gap/2⌋ > 1 AND ⌊gap/2⌋ > 0)]
⌊gap/2⌋-signo($C_{(0,0)}$)	[Oncoming AND gap ≤ 2*v_{max}-1 AND ⌊gap/2⌋ > 1 AND ⌊gap/2⌋ > 0]
$C_{(0,0)}$ – signo($C_{(0,0)}$)	[$\|C_{(0,0)}\| = v_{max}$ AND Oncoming AND NOT(gap ≤ 2*v_{max}-1) AND |$C_{(0,0)}$ |> 1 AND $C_{(0,0)}$ > 0] OR [$\|C_{(0,0)}\| = v_{max}$ AND NOT(Oncoming) AND NOT(|$C_{(0,0)}$| > gap) AND |$C_{(0,0)}$ |> 0 AND $C_{(0,0)}$> 1 AND brake]
gap*signo($C_{(0,0)}$)	[$\|C_{(0,0)}\| < v_{max}$ AND NOT(Oncoming) AND (|$C_{(0,0)}$|+1 > gap) AND NOT(gap*signo($C_{(0,0)}$) > 0 AND |gap*signo($C_{(0,0)}$)| > 1 AND brake)] OR [$\|C_{(0,0)}\| = v_{max}$ AND NOT(Oncoming) AND (|$C_{(0,0)}$| > gap) AND NOT(gap*signo($C_{(0,0)}$) > 0 AND |gap*signo($C_{(0,0)}$)| > 1 AND brake)]
gap*signo($C_{(0,0)}$)-signo($C_{(0,0)}$)	[$\|C_{(0,0)}\| < v_{max}$ AND NOT(Oncoming) AND |$C_{(0,0)}$|+1 > gap AND gap*signo($C_{(0,0)}$) > 0 AND |gap*signo($C_{(0,0)}$)| > 1 AND brake] OR [$\|C_{(0,0)}\| = v_{max}$ AND NOT(Oncoming) AND |$C_{(0,0)}$| > gap AND gap*signo($C_{(0,0)}$) > 0 AND |gap*signo($C_{(0,0)}$)| > 1 AND brake]

Donde
signo($C_{(0,0)}$), es una función que vale 1 si $C_{(0,0)}$>0 y vale 0 si $C_{(0,0)}$<0.

Oncoming equivale a la siguiente conjunción
(k es la primera posición hacia adelante ocupada por un auto AND
[($C_k < 0$ AND $C_{(0,0)} > 0$) OR ($C_k > 0$ AND $C_{(0,0)} < 0$)]).

Reglas actualizar la posición (paso 3)

Una vez efectuada la actualización de las velocidades correspondientes, en base al estado obtenido en ese paso, se procede a. actualizar las posiciones de los vehículos.

Nuevo Estado	Estado del vecindario
$C_{(k1,k2)}$	[$C_{(0,0)} = 0$ AND k=(k1,k2) es la primera posición hacia atrás ocupada por un auto AND $C_k > 0$ AND $C_{(k1,k2)}+k2 = 0$] OR [$C_{(0,0)} = 0$ AND k=(k1,k2) es la primera posición hacia adelante ocupada por un auto AND $C_k < 0$ AND $C_{(k1,k2)}+k2 = 0$]
0	[TodosVecinosHaciaAtrásVacías AND $C_{(0,0)} = 0$ AND TodosVecinosHaciaAdelanteVacías] OR [$C_{(0,0)} \neq 0$ AND $C_{(0,0)} \neq 1$ AND $C_{(0,0)} \neq -1$] OR [$C_{(0,0)} = 0$ AND k=(k1,k2) es la primera posición hacia atrás ocupada por un auto AND $C_k > 0$ AND $C_{(k1,k2)}+k2 \neq 0$] OR [$C_{(0,0)} = 0$ AND k=(k1,k2) es la primera posición hacia adelante ocupada por un auto AND $C_k < 0$ AND $C_{(k1,k2)}+k2 \neq 0$]
$C_{(0,0)}$	$C_{(0,0)} = 1$ OR $C_{(0,0)} = -1$

Donde,
TodosVecinosHaciaAtrásVacías, equivale a la siguiente conjunción:

$$\bigwedge_{i \in \{-v_{\max},\ldots,-1\}} C_{(0,i)} = 0$$

TodosVecinosHaciaAdelanteVacías, equivale a la siguiente conjunción:

$$\bigwedge_{i \in \{1,\ldots,v_{\max}\}} C_{(0,i)} = 0$$

“La primera posición hacia atrás ocupada por un auto”, es aquel vecino sobre el mismo carril ocupado por un auto con el número de índice máximo, menor a cero. Por ejemplo, si $C_{(0,-1)} = -1$ y $C_{(0,-2)} = -1$ pero $C_{(0,-3)} = 4$ entonces la primera posición hacia atrás ocupada por un auto es (0,-3).
“La primera posición hacia adelante ocupada por un auto”, es aquel vecino sobre el mismo carril ocupado por un auto con el número de índice mínimo, mayor a cero. Por ejemplo, si $C_{(0,1)} = -1$ y $C_{(0,2)} = -1$ pero $C_{(0,3)} = 4$ entonces la primera posición hacia adelante ocupada por un auto es (0,3).

“Todas las posiciones hacia atrás vacías”, equivale a la siguiente conjunción:

$$\bigwedge_{i \in \{-C_i.v_{\max},\ldots,-1\}} C_i \,.\mathrm{v}_{\text{actual}} = -1$$

B.2.2. **Modelos con cruce de carriles**

En sección representa los modelos con 2 carriles de distinta dirección e intersecciones, permitiendo que los vehículos puedan seguir derecho o doblar al llegar a un cruce.

B.2.2.i. **Modelos simples**

En [CQL95], [CLQ96] y [CDL97] se modela el tráfico sobre dos carriles con distinta dirección, sin permitir usar el carril de dirección opuesta para adelantarse. Luego, estos carriles se conectan con otros a través de los cruces.

Cada carril se modela a través de un *autómata celular* de una dimensión, donde el estado de las celdas representa la presencia o ausencia de un vehículo. Para definir el comportamiento de las celdas que no son adyacentes a los cruces, sólo se debe evaluar el estado de la celda anterior y de la siguiente. La regla de movimiento de los vehículos que se define es que un auto avanza a la siguiente posición siempre y cuando esa celda esté vacía; caso contrario permanece en su posición. El estado de la celda i en el paso de tiempo t+1 se calcula mediante la siguiente ecuación:

$$s_i(t+1) = s_{i-1}(t)(1- s_i(t)) + s_i(t)\, s_{i+1}(t)$$

Donde, $s_k(t)$ representa el estado de la celda k en el paso de tiempo t; éste puede valer 0 si está vacía, ó 1 si está ocupada por un vehículo.

Hasta el momento se ha definido el comportamiento de las celdas que se encuentran a lo largo de las calles, sin llegar a los cruces; faltaría ver cómo definir las intersecciones y cómo su presencia afecta a las celdas adyacentes sobre los carriles de entrada y salida.

Las intersecciones se representan como un anillo de celdas que tiene conectados varios carriles, como los descriptos anteriormente. Esto se muestra en la Figura 95.

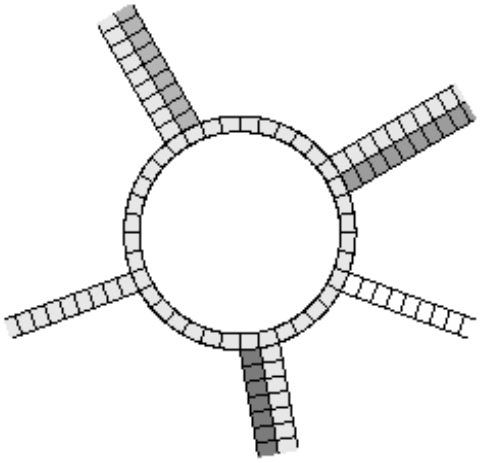


Figura 95 - Intersección de carriles

Las reglas de comportamiento de los vehículos establecen que un auto dentro de la intersección (en el anillo) tiene prioridad de acceso a una posición sobre cualquier otro vehículo que está fuera de ella. Además cada auto conoce por qué enlace desea salir, esto puede ser por una elección aleatoria al ingresar al cruce o por un plan de ruteo asociado al vehículo. Dentro de la intersección, un vehículo se mantiene girando en sentido contrario a las agujas del reloj o sale, es decir no se permite que permanezca en la misma posición porque esto puede provocar que en algún momento, nadie se pueda mover por estar esperando que otro vacíe la posición (deadlock). En cada paso de tiempo, cada vehículo se mueve a una celda adyacente vacía, ésta puede estar dentro de la intersección o ser la primera celda de un carril de salida del cruce.

Otra alternativa para modelar la elección del enlace de salida del cruce, puede ser tener un flag f, local a la celda. Cada vez que un auto pasa por una salida potencial de la intersección debe chequear el valor de f para determinar si seguir girando dentro ella o salir. El valor de f puede ser configurado para modelar distintas necesidades, puede ser constante por alguna cantidad de tiempo para imponer un movimiento particular en un cruce dado; puede ser completamente aleatoria, puede ser aleatoria sólo para algunos carriles y favorecer el tráfico en alguna dirección, o puede cambiar en forma determinística de acuerdo a alguna regla.

Este modelo puede permitir fácilmente la incorporación de semáforos, pues sólo hay que evitar que los vehículos ingresen a la intersección en determinados intervalos de tiempo. Similarmente, se pueden implementar otros modelo de prioridades.

Este modelo se puede representar como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

S = { 0,1 }

Donde,

0 representa la celda vacía,

1 representa que la celda tiene un vehículo.

n = 2

Este modelo presenta celdas con distinta vecindad y comportamiento, por un lado están las celdas dentro de las intersecciones, por otro las de ingreso a ellas y las de salida; y por último las celdas de los carriles que no son adyacentes a las intersecciones. Para la definición de las reglas se considera que un auto dentro del cruce siempre se puede mover, es decir, por lo menos su posición de adelante está siempre vacía.

1) Celdas dentro de las intersecciones

Las celdas que se hallan dentro del anillo, que modela los cruces, tienen dos celdas vecinas que corresponden a la posición anterior dentro del anillo y a la posición siguiente (en sentido contrario a las agujas del reloj). Se llamará C_0 a la celda cuyo comportamiento es descripto, $C_{(a,in)}$ y $C_{(a,out)}$ a la vecina de atrás y de adelante dentro de la intersección, respectivamente. Además, cada celda del anillo puede o no tener un vecino correspondiente a una entrada desde un carril, $C_{(c,in)}$, y/o a una salida hacia un carril, $C_{(c,out)}$.

En definitiva, existen 4 tipos de celdas de cruce, las que no tienen ni entradas ni salidas desde/hacia carriles, las que tienen una salida y sin entrada, las que tienen una entrada y sin salida; y por último, las que tienen una salida y una entrada

Celdas sin entradas y ni salidas

$\boldsymbol{\eta} = 3$

N = { 0, (a,in) , (a,out)}

La función $\boldsymbol{\tau}$ se define como:

Nuevo Estado	Estado del vecindario
0	($C_{(a,in)}$ = 0 AND C_0 = 0) OR C_0 = 1
1	($C_{(a,in)}$ = 1 AND C_0 = 0)

Celdas sin entradas y con una salida

$\eta = 4$

N = { 0, (a,in) , (a,out), (c,out)}

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	$(C_{(a,in)} = 0$ AND $C_0 = 0$) OR $C_0 = 1$
1	$(C_{(a,in)} = 1$ AND $C_0 = 0$)

Celdas con una entrada y sin salidas

$\eta = 4$

N = { 0, (a,in) , (a,out), (c,in)}

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	$(C_{(a,in)} = 0$ AND $C_0 = 0$ AND OR $C_{(c,in)} = 0)$ OR $C_0 = 1$
1	$(C_{(a,in)} = 1$ OR $C_{(c,in)} = 1)$ AND $C_0 = 0$

Celdas con una entrada y una salida

$\eta = 5$

N = { 0, (a,in) , (a,out), (c,in), (c,out)}

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	$(C_{(a,in)} = 0$ AND $C_0 = 0$ AND OR $C_{(c,in)} = 0)$ OR $C_0 = 1$
1	$(C_{(a,in)} = 1$ OR $C_{(c,in)} = 1)$ AND $C_0 = 0$

2) Celdas de ingreso a las intersecciones

Ahora se define el comportamiento de las celdas que pertenecen a algún carril y son adyacentes a una intersección. Se llamará C_0 a la celda cuyo comportamiento es descripto, C_{in} a la vecina de atrás del mismo carril, C_{out} a la de adelante dentro del la intersección; y $C_{(a,in)}$ a la celda del anillo de entrada para la vecina C_{out}.

$\eta = 4$

N = { 0, in, out, (a,in)}

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	(C_{in} = 0 AND C_0 = 0) OR (C_0 = 1 AND C_{out} = 0 AND $C_{(a, in)}$ = 0)
1	(C_{in} = 1 AND C_0 = 0) OR (C_0 = 1 AND (C_{out} = 1 OR $C_{(a, in)}$ = 1))

3) Celdas de salida de las intersecciones

A continuación se describe el comportamiento de las celdas que pertenecen a algún carril de salida de una intersección y son vecinas de alguna celda del anillo. Se llamará C_0 a la celda cuyo comportamiento es descripto, C_{in} a la vecina de atrás dentro de la intersección y C_{out} a la de adelante del mismo carril.

$\eta = 3$

N = { 0, in, out}

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	(C_{in} = 0 AND C_0 = 0) OR (C_0 = 1 AND C_{out} = 0) OR (C_{in} = 1 AND C_0 = 0 AND NOT(salir))
1	(C_{in} = 1 AND C_0 = 0 AND salir) OR (C_0 = 1 AND C_{out} = 1)

Donde,

salir, es una función booleana local a la celda, que determina si el vehículo debe tomar esa salida o seguir en la intersección. Si salir es verdadero, el vehículo sale de la intersección si la celda de destino está vacía, caso contrario permanece girando en el anillo.

4) Celdas de los carriles que no son adyacentes a ninguna celda de la intersección

Se llamará C_0 a la celda cuyo comportamiento es descripto, C_{in} y C_{out} a la vecina de atrás y de adelante del mismo carril.

$\eta = 3$

N = { 0, in, out}

La función τ se define como:

Nuevo Estado	Estado del vecindario
0	(C_{in} = 0 AND C_0 = 0) OR (C_0 = 1 AND C_{out} = 0)
1	(C_{in} = 1 AND C_0 = 0) OR (C_0 = 1 AND C_{out} = 1)

C. **Modelos de tráfico de más de dos carriles**

Aquí se incluyen los modelos de varios carriles que pueden ser de igual o distinta dirección. En general son extensiones de otros con menor cantidad de carriles, pero al ampliar el vecindario

deben tener mayor cuidado con conflictos de acceso a las mismas posiciones por diferentes vehículos.

C.1. **Modelos con una única dirección**

En estos modelos todos los carriles tienen la misma dirección y las reglas deben considerar los movimientos de cambio de carril, que pueden ser representados mediante reglas simétricas o asimétricas.

C.1.1. **Modelos con reglas simétricas**

Estos modelos tienen varios carriles de igual dirección y utilizan las mismas reglas para decidir el cambio de carril hacia derecha o izquierda.

C.1.1.1. **Modelos con cruce de carriles**

En esta categoría se ubican los modelos de varios carriles de igual dirección, con reglas siméricas que permiten la definición de cruces.

C.1.1.1.i. **Modelos con vehículos especiales (autos con ruteo), controles tránsito en los cruces (señales, reglas predefinidas)**

En [NSPLDB98] se plantea un modelo llamado TRANSIMS (**Tr**ansportation **An**alysis **Sim**ulation **S**ystem) que fue diseñado para considerar el *planeamiento* del transporte, es decir, permitir que los vehículos puedan decidir qué camino tomar para llegar a destino. En los modelos anteriores, los autos se mueven siempre en la misma dirección, o bien, giran en forma no determinística; a diferencia de TRANSIMS, donde los vehículos doblan en los enlaces para seguir su ruta a destino.

El modelo multicarril se construye en forma incremental, partiendo del modelo para el tráfico en *un carril unidireccional* descripto en la sección A.1 definido en [NS92]. Éste es extendido para el tráfico de *dos carriles unidireccionales* realizando la actualización del estado de los vehículos del sistema en dos pasos:

1. *Decisión de cambio de carril*.
2. *Actualización según el modelo de un carril.*

Para tomar la decisión del cambio de carril se deben verificar las siguientes condiciones:

IF x_o está vacía
THEN IF (gap < v AND gap_o > gap)
THEN weight1 = 1
ELSE weight1 = 0
weight2 = v - gap_o
weight3 = v_{max} - $gap_{o,back}$
IF (weight1 > weight2) AND (weight1 > weight3)
THEN cambiar de carril con probabilidad p_{change}.

Donde, x_o es la posición de al lado de la posición actual del vehículo. p_{change} se utiliza para evitar el efecto ping pong, frenando en forma no determinística con esa probabilidad, al igual que en los modelos anteriores. Las variables weight se utilizan para permitir incorporar la planificación de recorridos.

Finalmente, el modelo anterior es extendido para el tráfico de *tres o más carriles unidireccionales*, agregando una restricción sobre la decisión del cambio de carril. Esto es necesario porque cuando hay más de 2 carriles con las reglas anteriores se pueden producir colisiones; por ejemplo, en una calle de 3 carriles, un vehículo del carril izquierdo y otro del

derecho deciden pasar a la misma posición del carril del medio. Para evitar estos problemas se propone la siguiente política de actualización de posiciones:

IF el paso de tiempo es par
THEN evaluar el cambio de carril hacia la izquierda para los autos del medio o de la derecha.
IF el paso de tiempo es impar
THEN evaluar el cambio de carril hacia la derecha para los autos del medio o del carril izquierdo.

Teniendo definido el modelo para varios carriles, se agregan al mismo características que permitan representar el movimiento de los vehículos dirigidos hacia algún destino particular, realizando cambios de carril que le permitan seguir un plan o camino determinado. Este *plan de ruteo* está formado por la secuencia de autopistas o enlaces que deben tomar para llegar a destino. De esta forma cuando se aproximan a una intersección, deben moverse al carril correcto para poder tomar el desvío pretendido. Por ejemplo, si un auto debe girar a la izquierda en la siguiente intersección, debe pasar a alguno de los carriles que permiten realizar ese giro. Para poder lograr los giros que le indican los planes, se agregan a las reglas anteriores otro peso, de la siguiente manera:

IF (weight1 + weight4 > weight2) AND (weight1 + weight4 > weight3)
THEN cambiar de carril con probabilidad p_{change}.

donde,

$$weight4 = \max\left(\frac{d^* - d}{v_{\max}}, 0\right)$$

donde, d es la cantidad de posiciones que faltan hasta la intersección y d* es una cantidad de posiciones, dada como parámetro.

El peso weight4 aumenta desde 0 hasta d^*/v_{max} a medida que el vehículo se acerca a la intersección y sus distintos valores provocan un comportamiento diferente sobre el vehículo. Cuando weight4 vale 0, entonces no influencia la decisión de cambiar de carril. Cuando weight4 vale 1, provoca el mismo efecto que si hubiera un auto lento adelante en el mismo carril. Y así, a medida que aumenta weigth4, se violan los criterios de seguridad de la distancia hacia adelante y atrás que debe existir en el carril destino para poder cambiar de carril sin estorbar el camino de otro auto. Hasta que cuando weigth4 supera a v_{max}, entonces permite que el vehículo haga el cambio de carril a pesar que sólo la celda vecina en el carril destino esté libre.

Una vez que el vehículo está en el carril "correcto" a menos de d* celdas de la intersección, sólo se permite que el vehículo cambie a otro carril si éste también es "correcto". Cuando hay varios carriles habilitados para acceder a una intersección, por lo anterior, el uso de esos carriles es equivalente.

Para que los vehículos puedan seguir el plan de ruteo, deben realizar giros que les permitan tomar los enlaces necesarios para alcanzar el destino. Estos giros pueden ser desprotegidos, es decir, un vehículo debe atravesar otros carriles que tienen *prioridad*. Este es el caso de las señales de STOP y Ceda el Paso, rampas de entradas a autopistas, giros a la izquierda desprotegidos (esto se da, por ejemplo, cuando se intenta un giro a la izquierda para ingresar en una calle doble mano. Los carriles con privilegio serían los que vienen de la izquierda y el de más a la izquierda de los que vienen de la derecha). Para modelar estos movimientos se utiliza un "espacio de aceptación", que corresponde a una cantidad de posiciones que deben estar vacías en los carriles con privilegio, y para TRANSIMS se utiliza un tamaño equivalente a 3 veces la velocidad del primer auto que viene por el respectivo carril privilegiado. Es decir, el movimiento es aceptado si existe ese espacio entre la intersección y el vehículo que viene en sentido opuesto.

Luego se establecen las reglas que permiten modelar dos tipos de *intersecciones*, señalizadas y no señalizadas. Las primeras son aquellas donde las prioridades cambian con el tiempo y son reguladas por señales. Mientras que las no señalizadas, tienen prioridades fijas.

Cuando un vehículo simulado se acerca a una *intersección señalizada*, el algoritmo primero decide si potencialmente quiere dejar el enlace o autopista, observando su velocidad. Si el auto quiere salir del enlace, el algoritmo "chequea el control del tráfico", que es lo que determina si el vehículo puede o no dejar el enlace. Se pueden dar los siguientes casos:

- señal de prohibición (luz roja), entonces el vehículo no puede abandonar el enlace y no se realiza ninguna otra acción.
- señal de protección (flecha verde) o alerta (luz amarilla), entonces el vehículo puede entrar a la intersección.
- señal de permiso (luz verde), entonces el vehículo debe chequear que haya suficiente espacio en todos los flujos que tengan privilegio para esta maniobra.

Si se acepta el *ingreso a la intersección*, el auto se mueve a una "cola de la intersección"; cada intersección tiene una cola por carril entrante. Las colas modelan el comportamiento del vehículo dentro de la intersección. Cuando un auto ingresa, recibe un "time stamp", antes del cual no se permite que salga de la intersección y representa la duración del movimiento en su interior. Las colas de las intersecciones tienen capacidad finita y cuando se llenan no permiten que ingresen más vehículos, teniendo que esperar dentro del enlace para ingresar.
Cuando un vehículo está listo para *salir de la intersección*, se intenta mover a la primera celda del enlace de destino, si está vacía. La velocidad con que sale es la misma con la que tenía al ingresar a la intersección.
Cuando los vehículos giran sobre el tráfico opuesto, toman la decisión de aceptar el giro cuando ingresan a la cola de la intersección; no cuando salen. Esto puede provocar que un vehículo que entró a la intersección cuando no venía tráfico de frente, no haga la maniobra en forma inmediata por la presencia de otros autos en la cola. El giro se hará cuando la cola esté vacía y haya una celda vacía en el enlace destino. Esto puede producir que el auto gire a través del tráfico que se acerca, pero en promedio se observa el comportamiento correcto.

Las *intersecciones no señalizadas* no tienen colas internas (los vehículos las atraviesan directamente) y los autos que salen de ellas, avanzan sobre el nuevo enlance tan rápido como su velocidad lo indique (no ingresan necesariamente, en la primera posición del mismo). Estas son las diferencias que tienen con las señalizadas, salvo por ello, sus comportamientos son similares.
Cuando un vehículo simulado se acerca a una intersección sin señalización, el algoritmo primero decide si potencialmente quiere dejar el enlace o autopista, observando su velocidad. Si el auto quiere salir del enlace, el algoritmo "chequea el control del tráfico", que es lo que determina si el vehículo puede o no dejar el enlace.
Los controles de tráfico pueden ser:

1. "Sin Control": en este caso, el vehículo se mueve a la celda de destino sin ningún otro chequeo. Se utiliza en general, para los carriles que tienen prioridad. Si un vehículo tiene velocidad 5 y quedan sólo 2 celdas en su enlace, entonces trata de avanzar 3 posiciones en el carril de destino. Si esa posición está ocupada se moverá a la más cercana a la misma que esté vacía.
2. "Ceda el Paso": en este caso, el vehículo chequea el espacio en todos los carriles opuestos , de acuerdo al "espacio de aceptación". Si el movimiento es aceptado, la celda de destino se elige de acuerdo a las reglas del caso "Sin Control".
3. "Señal de STOP": sólo cuando el vehículo tiene velocidad cero por al menos un paso de tiempo en la última celda del enlace, se permite que continúe. Luego de atravesar la intersección, debe existir un espacio de 3 veces la velocidad de los autos que se acercan en cada carril opuesto. Si el movimiento es aceptado, el vehículo que atravesó la señal de STOP, irá a la primera celda del enlace de destino (si está vacía) y tendrá velocidad 1.

El modelo permite el ingreso y salida de los vehículos de la simulación utilizando los *estacionamientos* que es dónde comienzan y terminan los viajes de los vehículos. Cada enlace

de la microsimulación (salvo por las rampas de ascenso y descenso de las autopistas, enlaces entre autopistas) tienen por lo menos un estacionamiento. Cada vehículo tiene un *plan de ruteo* y el *tiempo de comienzo* del viaje, en ese tiempo de comienzo, el vehículo es agregado en una cola de autos que quieren salir del estacionamiento. Cuando el vehículo llega al principio de la cola, intenta ingresar en el enlace si existe un espacio vacío y cuando un vehículo llega a su estacionamiento de destino, de acuerdo a su plan, saldrá de la simulación.

Finalmente, hay que establecer un orden de ejecución de las reglas que modelan los distintos aspectos descriptos. Para ello se definen los siguientes subpasos que deben ser cumplidos en orden para la actualización del estado del sistema. Sea t el tiempo en que se realizó la última actualización, sean t_1,t_2,.., los tiempos parciales, entonces:

1. Los vehículos que están listos para dejar las colas de las *intersecciones señalizadas* reservan celdas en los carriles de salida. Ellos sólo intentan reservar la primera posición del enlace. Si esa celda está ocupada (por la reservación de otro auto) entonces no puede salir de la intersección. Pueden darse conflictos entre varias colas para la misma posición. Esto se resuelve asignando orden FCFS (First Come First Served) a las colas para ser atendidas y las primeras tendrán chance de vaciarse más rápidamente. El resultado de este paso es la información obtenida en t_1.
2. Los vehículos *cambian de carril*, usando la información calculada en t_1. El resultado de este paso es la información obtenida en t_2.
3. Se realizan las *salidas de los estacionamientos*, cuyo resultado se obtiene en t_3.
4. Los vehículos que quieren atravesar las *intersecciones no señalizadas* reservan celdas en los carriles de destino. Pueden darse conflictos entre varias carriles entrantes que compiten por la misma posición de salida. Esto se resuelve asignando orden FCFS (First Come First Served) a los carriles para ser atendidos y los primeros tendrán la chance de vaciarse más rápidamente. Esto sólo sucede con los carriles secundarios pues los enlaces principales nunca compiten por el mismo enlace de salida. El resultado de este paso es la información obtenida en t_4.
5. Se calculan las *velocidades* de los autos y se actualizan las *posiciones*. Si un vehículo que tenía planeado atravesar la intersección, no lo hace por el resultado de la actualización de la velocidad, la reservación es cancelada. Al atravesar intersecciones no señalizadas donde el resultado de la actualización de la velocidad los hace ingresar en la misma, se dirigen a la celda reservada anteriormente. El resultado de este paso es la información obtenida en t_5.

Lo que se obtiene en t_5 equivale a la actualización del paso t+1.

Este modelo se puede representar como:

$$CCA = < S, n, C, \eta, N, T, \tau, c.\boldsymbol{Z}_0^+ >$$

S = { (v_{actual}, plan) / $v_{actual} \in \{-2, -1, 0, 1, 2, ..., v_{max}\}$ y plan es un vector de posiciones enteras }

Donde,

v_{actual} = -2 representa que la celda fue reservada por un auto que intenta cruzar una intersección

v_{actual} = -1 representa que la celda está vacía,

v_{actual} = 0 representa la presencia de un auto detenido en la celda,

v_{actual} = 1 representa la presencia de un auto con velocidad 1,

y así siguiendo.

plan(i) indica cuál es el enlace que hay que tomar en la i_ésima intersección.

n = 2

Este modelo presenta distintos tipos de celdas con vecindad y comportamiento diferente. Por un lado, las celdas "comunes", son aquellas que no están en contacto con las intersecciones, es decir, desde de ellas no se puede alcanzar una intersección en una sola actualización de estados. Éstas tampoco tienen cercanía a los estacionamientos. Luego están las celdas cercanas a una intersección señalizada, las cercanas a una intersección no señalizada y las cercanas a los estacionamientos.

Celdas comunes

$\eta = 6^* v_{max} + 3$

N = VecinosHaciaAdelante U VecinosHaciaAtrás U VecinosHaciaAdelanteIzquierda U VecinosHaciaAdelanteDerecha U VecinosHaciaAtrásIzquierda U VecinosHaciaAtrásDerecha U { (0,0); (1,0); (-1,0) }

Donde,

VecinosHaciaAdelante = {(0,1); (0,2); (0,3);...; (0, v_{max})}
VecinosHaciaAtrás = {(0,-1); (0,-2); (0,-3);...; (0,- v_{max})}
VecinosHaciaAdelanteIzquierda = {(1,1); (1,2); (1,3);...; (1, v_{max})}
VecinosHaciaAdelanteDerecha = {(-1,1); (-1,2); (-1,3);...; (-1, v_{max})}
VecinosHaciaAtrásIzquierda = {(1,-1); (1,-2); (1,-3);...; (1, -v_{max})}
VecinosHaciaAtrásDerecha = {(-1,-1); (-1,-2); (-1,-3);...; (-1,- v_{max})}

(1,-v_{max})	...	(1,-1)	(1,0)	(1,1)	...	(1, v_{max})		
(0,-v_{max})	...	(0,-1)	(0,0)	(0,1)	...	(0, v_{max})		
(-1,-v_{max})	...	(-1,-1)	(-1,0)	(-1,1)	...	(-1, v_{max})		

Figura 96 - Vecindario de la celda origen

La función τ para estas celdas se calcula en los subpasos de tiempo t2 y t5; en el primero se efectúan los cambios de carril (sin avanzar el vehículo), en el segundo se actualiza la velocidad y posición de acuerdo de cada vehículo.

Reglas para cambio de carril (subpaso t2)

Caso 1: [σ(t) AND $C_{(0,0)}.v_{actual}$ >-1 AND HaciaIz AND $C_{(1,0)}.v_{actual}$ = -1]
Si gap(0,0)< $C_{(0,0)}.v_{actual}$ AND $gap_o(0,0)$ > gap
Entonces w1=1
Sino w1=0
w2= $C_{(0,0)}.v_{actual}$ > $gap_o(0,0)$
w3= v_{max} > $gap_{o,back}(0,0)$

$$w4 = \max\left(\frac{d^* - d_{\text{int}\,er\,\text{sec}}(0,0)}{v_{\max}}, 0\right)$$

Si (w1+w4>w2) AND (w1+w4>w3) AND change
Entonces $C_{(0,0)}.v_{actual}$ = -1
Sino $C_{(0,0)}$= $C_{(0,0)}$

gap_o se refiere al carril izquierdo.

Caso 2: [NOT(σ(t)) AND $C_{(0,0)}.v_{actual}$ >-1 AND HaciaDer AND $C_{(-1,0)}.v_{actual}$ = -1]

Si gap(0,0)< $C_{(0,0)}.v_{actual}$ AND $gap_o(0,0)$ > gap
Entonces w1=1
Sino w1=0
w2= $C_{(0,0)}.v_{actual}$ > $gap_o(0,0)$
w3= v_{max} > $gap_{o,back}(0,0)$

$$w4 = \max\left(\frac{d^* - d_{\text{int er sec}}(0,0)}{v_{\max}}, 0\right)$$

Si (w1+w4>w2) AND (w1+w4>w3) AND change
Entonces $C_{(0,0)}.v_{actual}$ = -1
Sino $C_{(0,0)}$= $C_{(0,0)}$

gap_o se refiere al carril derecho.

Caso 3: [(NOT(σ(t)) OR $C_{(1,0)}.v_{actual}$ > -1) AND $C_{(0,0)}.v_{actual}$ >-1 AND HaciaIZ]
Entonces $C_{(0,0)}$= $C_{(0,0)}$

Caso 4: [(σ(t) OR $C_{(-1,0)}.v_{actual}$ > -1) AND $C_{(0,0)}.v_{actual}$ >-1 AND HaciaDer]
Entonces $C_{(0,0)}$= $C_{(0,0)}$

Caso 5: [σ(t) AND $C_{(-1,0)}.v_{actual}$ >-1 AND HaciaIz(-1,0) AND $C_{(0,0)}.v_{actual}$ = -1]
Si gap(-1,0)< $C_{(-1,0)}.v_{actual}$ AND $gap_o(-1,0)$ > gap(-1,0)
Entonces w1=1
Sino w1=0
w2= $C_{(-1,0)}.v_{actual}$ > $gap_o(-1,0)$
w3= v_{max} > $gap_{o,back}(-1,0)$

$$w4 = \max\left(\frac{d^* - d_{\text{int er sec}}(-1,0)}{v_{\max}}, 0\right)$$

Si (w1+w4>w2) AND (w1+w4>w3) AND change
Entonces $C_{(0,0)}.v_{actual}$ = $C_{(-1,0)}$
Sino $C_{(0,0)}$= $C_{(0,0)}$

Caso 6: [NOT(σ(t)) AND $C_{(1,0)}.v_{actual}$ >-1 AND HaciaDer(1,0) AND $C_{(0,0)}.v_{actual}$ = -1]
Si gap(1,0)< $C_{(1,0)}.v_{actual}$ AND $gap_o(1,0)$ > gap(1,0)
Entonces w1=1
Sino w1=0
w2= $C_{(1,0)}.v_{actual}$ > $gap_o(1,0)$
w3= v_{max} > $gap_{o,back}(1,0)$

$$w4 = \max\left(\frac{d^* - d_{\text{int er sec}}(1,0)}{v_{\max}}, 0\right)$$

Si (w1+w4>w2) AND (w1+w4>w3) AND change
Entonces $C_{(0,0)}.v_{actual}$ = $C_{(1,0)}$
Sino $C_{(0,0)}$= $C_{(0,0)}$

Caso 7: [NOT(σ(t)) AND $C_{(-1,0)}.v_{actual}$ > -1 AND HaciaIZ]
Entonces $C_{(0,0)}$= $C_{(0,0)}$

Caso 8: [σ(t) AND $C_{(1,0)}.v_{actual}$ > -1 AND HaciaDer]
Entonces $C_{(0,0)}$= $C_{(0,0)}$

OtroCaso :
Entonces $C_{(0,0)}$= $C_{(0,0)}$

Reglas para movimiento de avance (subpaso t5)

Las reglas para este paso, son las mismas del modelo de carril único de [NS92] y ya fueron definidas en la sección A.1. No es necesario que sean modificadas pues son aplicadas sobre cada carril en forma totalmente independiente, una vez que se realizaron los cambios de carril correspondientes.

Celdas cercanas a las intersecciones

Las celdas ubicadas cerca de una intersección, es decir, aquellas que están a menos de d* posiciones o luego de d* posiciones de la misma; deben tener ciertos datos a cerca del cruce que se mantienen constantes durante toda la simulación. Estas celdas deben saber qué tipo de intersección tienen cerca y a qué distancia están de la misma ($d_{intersec}$). El tipo de intersección incluye conocer si es señalizada o no y cómo determinar si se puede cruzar o no en ese instante.

Estas celdas pueden dividirse en celdas cercanas a una intersección señalizada y celdas cercanas a una intersección no señalizada. A su vez cada uno de esos grupos se puede dividir en las celdas que están ubicadas antes del cruce y luego del cruce, considerando comportamientos distintos en cada caso.

a) Celdas cercanas a intersecciones señalizadas

Celdas ubicadas antes de la intersección

Las reglas de actualización de los estados de estas celdas se aplican en los subpasos de tiempo t2 y t5; en el primero se efectúan los cambios de carril (sin avanzar el vehículo), en el segundo se actualiza la velocidad y posición de cada vehículo.

En t2, si la celda estaba vacía ($C_{(0,0)}.v_{actual} = -1$) y algún auto de la posición de al lado hizo el cambio de carril hacia ella, entonces su nuevo estado es ocupada por ese vehículo. En otro caso permanece vacía. Si en t2, la celda estaba ocupada, puede vaciarse si se dan las condiciones para un cambio de carril (éste es posible si el vehículo cambia hacia otro "carril correcto"). En otro caso permanece ocupada.

En t5, si la celda estaba vacía y algún auto de atrás actualizó su posición justo en ella, entonces su nuevo estado es ocupada por ese vehículo. En otro caso permanece vacía. Si en t5, la celda estaba ocupada, puede vaciarse si el auto logra ingresar a la intersección (se debe controlar la luz del semáforo y si hay lugar en ella) o avanza unas posiciones. En otro caso permanece ocupada.

Celdas ubicadas luego de la intersección

Las reglas de actualización de los estados de estas celdas se aplican en los subpasos de tiempo t1, t2 y t5; en el primero se efectúan las reservaciones para los vehículos que salen de la intersección, en el segundo se efectúan los cambios de carril (sin avanzar el vehículo), en el tercero se actualiza la velocidad y posición de cada vehículo.

En t1, si la celda estaba vacía ($C_{(0,0)}.v_{actual} = -1$), pasa a estar reservada si algún auto de la intersección quiere salir hacia ella. En otro caso permanece vacía.
Si en t1, la celda estaba ocupada, permanece ocupada.

En t2, si la celda estaba vacía y algún auto de la posición de al lado hizo el cambio de carril hacia ella, entonces su nuevo estado es ocupada por ese vehículo. En otro caso permanece vacía.
Si en t2, la celda estaba ocupada, puede vaciarse si se dan las condiciones para un cambio de carril (éste es posible si el vehículo cambia hacia otro "carril correcto"). En otro caso permanece ocupada.

Si en t2, la celda estaba reservada, permanece reservada.

En t5, si la celda estaba vacía y algún auto de atrás actualizó su posición justo en ella, entonces su nuevo estado es ocupada por ese vehículo. En otro caso permanece vacía.
Si en t5, la celda estaba ocupada, puede vaciarse si el auto logra ingresar a la intersección (se debe controlar la luz del semáforo y si hay lugar en ella) o avanza unas posiciones. En otro caso permanece ocupada.
Si en t5, la celda estaba reservada, puede pasar a vacía si el auto que hizo la reservación no logra salir de la intersección o pasa a estar ocupada por ese vehículo.

Hay que tener en cuenta que la reservación de las celdas de salida de la intersección se realiza de acuerdo al plan del vehículo ($C_{(i,j)}$.plan), éste es el que indica qué enlace debe ser el de salida.

b) Celdas cercanas a intersecciones no señalizadas

Celdas ubicadas antes de la intersección

Las reglas de actualización de los estados de estas celdas se aplican en los subpasos de tiempo t2 y t5; en el primero se efectúan los cambios de carril (sin avanzar el vehículo), en el segundo se actualiza la velocidad y posición de cada vehículo.
En t2, si la celda estaba vacía ($C_{(0,0)}.v_{actual} = -1$) y algún auto de la posición de al lado hizo el cambio de carril hacia ella, entonces su nuevo estado es ocupada. En otro caso permanece vacía.
Si en t2, la celda estaba ocupada, puede vaciarse si se dan las condiciones para un cambio de carril (éste es posible si el vehículo cambia hacia otro "carril correcto"). En otro caso permanece ocupada.
En t5, si la celda estaba vacía y algún auto de atrás actualizó su posición justo en ella, entonces su nuevo estado es ocupada por ese vehículo. En otro caso permanece vacía. Si en t5, la celda estaba ocupada, puede vaciarse si el auto logra cruzar la intersección (se debe controlar el espacio de aceptación y que se haya podido reservar una posición en el enlace correspondiente) o avanza unas posiciones. En otro caso permanece ocupada.

Celdas ubicadas luego de la intersección

Las reglas de actualización de los estados de estas celdas se aplican en los subpasos de tiempo t4, t2 y t5; en el primero se efectúan las reservaciones para los vehículos que cruzan la intersección, en el segundo se efectúan los cambios de carril (sin avanzar el vehículo), en el tercero se actualiza la velocidad y posición de cada vehículo.

En t2, si la celda estaba vacía y algún auto de la posición de al lado hizo el cambio de carril hacia ella, entonces su nuevo estado es ocupada. En otro caso permanece vacía.
Si en t2, la celda estaba ocupada, puede vaciarse si se dan las condiciones para un cambio de carril (éste es posible si el vehículo cambia hacia otro "carril correcto"). En otro caso permanece ocupada.
Si en t2, la celda estaba reservada, permanece reservada.

En t4, si la celda estaba vacía ($C_{(0,0)}.v_{actual} = -1$), pasa a estar reservada si algún auto quiere atravesar la intersección hacia ella. En otro caso permanece vacía.
Si en t4, la celda estaba ocupada, permanece ocupada.

En t5, si la celda estaba vacía y algún auto de atrás actualizó su posición justo en ella, entonces su nuevo estado es ocupada. En otro caso permanece vacía.
Si en t5, la celda estaba ocupada, puede vaciarse si el auto logra atravesar la intersección. En otro caso permanece ocupada.
Si en t5, la celda estaba reservada, puede pasar a vacía si el auto que hizo la reservación no logra cruzar la intersección o pasa a estar ocupada.

Hay que tener en cuenta que la reservación de las celdas de salida de la intersección se realiza de acuerdo al plan del vehículo ($C_{(i,j)}$.plan), éste es el que indica qué enlace debe ser el de salida.

Celdas cercanas a los estacionamientos

Las reglas de actualización de los estados de estas celdas se aplican en los subpasos de tiempo t3, t2 y t5; en el primero se efectúan las salidas de lo estacionamientos, en el segundo se efectúan los cambios de carril (sin avanzar el vehículo), en el tercero se actualiza la velocidad y posición de cada vehículo.

En t2, si la celda estaba vacía y algún auto de la posición de al lado hizo el cambio de carril hacia ella, entonces su nuevo estado es ocupada. En otro caso permanece vacía.
Si en t2, la celda estaba ocupada, puede vaciarse si se dan las condiciones para un cambio de carril (éste es posible si el vehículo cambia hacia otro "carril correcto"). En otro caso permanece ocupada.
Si en t2, la celda estaba reservada, permanece reservada.

En t3, si la celda estaba vacía ($C_{(0,0)}.v_{actual} = -1$), pasa a estar ocupada si algún auto sale del estacionamiento cercano. En otro caso permanece vacía.
Si en t3, la celda estaba ocupada, permanece ocupada.

En t5, si la celda estaba vacía y algún auto de atrás actualizó su posición justo en ella, entonces su nuevo estado es ocupada. En otro caso permanece vacía.
Si en t5, la celda estaba ocupada, puede vaciarse si el auto logra atravesar la intersección. En otro caso permanece ocupada.
Si en t5, la celda estaba reservada, puede pasar a vacía si el auto que hizo la reservación no logra cruzar la intersección o pasa a estar ocupada.

6.3. Discusión

Al clasificar algunos de los modelos de autómatas celulares existentes para simulación de tráfico de vehículos se observa que la mayoría plantea el problema para 1 ó 2 carriles. Así, se logran reglas de comportamiento simples, que en algunos casos son extendidas para modelar otros aspectos. Pero a medida que se busca una solución más genérica, por ejemplo que represente cualquier cantidad de carriles, se incrementa la complejidad del modelo.

Otra simplificación, respecto a la representación del modelo, es que se definen autómatas celulares con bordes conectados; condicionando en cierta medida al flujo de vehículos. Mientras que sólo algunos, representan el ingreso y salida de autos de la simulación sin esta restricción.

El comportamiento modelado, en general, consiste del avance recto, cambio de carril o maniobras de adelantamiento determinados por el espacio existente respecto a los autos cercanos, representando variaciones de velocidad en algunos casos. Pero muy pocos incorporan otros aspectos que condicionan el flujo de vehículos, como ser la presencia de choques, obras, baches, vías de tren, etc. Más aún, de la segunda clasificación presentada en la sección 6.1 sólo se modelaron semáforos (secciones A.1.ii y A.2.ii), vehículos especiales (secciones A.1.iii, B.1.2.1.ii y C.1.1.1.i) y señales de control (sección C.1.1.1.i).

El modelado de algunos aspectos tales como los semáforos (sección A.2.ii) o la extensión de las reglas de 2 carriles a 3 (sección C.1.1.1.i), restringiendo los movimientos de acuerdo a los ciclos pares o impares del reloj de simulación; no surgen naturalmente del sistema real. Pero esta

forma de representación incorpora nuevas características sin modificar las reglas de comportamiento, sólo restringe el momento en que pueden ser aplicadas. Tal vez sería interesante analizar en cuánto aumenta la complejidad del modelo si se agregan estas características en forma más cercana al problema que a la implementación de la simulación.

A pesar de todas estas simplificaciones que se plantean en los modelos, la mayoría ha sido implementado, generando un comportamiento comparable con datos reales. Pero resulta interesante evaluar la precisión de los resultados versus la complejidad del modelo.

7. Formalismos de Especificación

En esta sección se incluye una definición de los paradigmas de especificación DEVS y Cell_DEVS como fueran presentados en [WG98]. Luego, en el capítulo II, serán utilizados para modelar diversas características del tráfico urbano.

7.1. DEVS

DEVS (Discrete EVents dynamic Systems) es un mecanismo de simulación jerárquica propuesto por Zeigler [Zei76]. Establece una teoría de modelado de sistemas a tiempo continuo usando modelado de eventos discretos. El paradigma provee una forma de especificar un objeto matemático llamado sistema. Éste se describe como un conjunto consistente de una base de tiempo, entradas, salidas y funciones para calcular los siguientes estados y salidas.

El formalismo define cómo generar nuevos valores para las variables y los momentos en los que estos valores deben cambiar. Puede verse como una forma de especificar sistemas cuyas entradas, estados y salidas son constantes de a trozos, y cuyas transiciones se identifican como eventos discretos.

Un modelo DEVS se construye en base a un conjunto de modelos básicos, que se combinan para formar modelos acoplados. Los modelos pueden ser de comportamiento (atómicos), o estructurales (acoplados). Un modelo acoplado especifica cómo se conectan las entradas y salidas de los componentes. Los nuevos modelos también son modelos básicos modulares, y pueden usarse para armar modelos de mayor nivel.

Un *modelo atómico* es una especificación de un modelo comportamental. Formalmente, esta especificación puede definirse como:

$M = <I, X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D>$

I = $< P^X, P^Y >$ representa la definición de la interfaz modular del modelo. En este caso, $\forall\ j \in [1, \eta]$, $i \in \{X, Y\}$, P_j^i es una definición de un port (de entrada o salida respectivamente), donde

$P_j^i = \{ (N_j^i, T_j^i)\ /\ \forall\ j \in [1, \mu], (\mu \in \boldsymbol{N}, \mu < \infty), N_j^i \in [i_1, i_m]$ (nombre del port), y T_j^i = Tipo del port$\}$;

X es el conjunto de eventos externos de entrada;

S es el conjunto de estados secuenciales;

Y es el conjunto de eventos externos generados para salida;

δ_{int}: $S \rightarrow S$ es la función de transición interna, que define los cambios de estado por causa de eventos internos;

δ_{ext}: $Q \times X \rightarrow S$ es la función de transición externa, que define los cambios de estado por causa de eventos externos. Q es el conjunto de estados totales del sistema especificado como $Q = \{ (s,e) / s \in S, e \in [0, D(s)] \}$, donde **e** representa el tiempo transcurrido desde la última transición de estado con estado **s**;

λ: $S \rightarrow Y$ es la función de salida;

D: S→$\boldsymbol{R}^+$ es la función de duración de un estado, donde D(s) es el tiempo que el modelo se queda en el estado s si no hay un evento externo.

Para especificar modelos DEVS es conveniente ver al modelo como con ports de entrada/salida que interactúan con el entorno (**I**). Cuando se reciben en los ports de entrada los eventos externos, la descripción del modelo debe determinar cómo responder.

Un modelo atómico DEVS transita entre los estados (**S**) vía sus funciones de transición. Si no hay eventos externos, los estados pueden cambiar ejecutando la función de transición interna (δ_{int}), cuya activación está determinada por la función de pasaje de tiempo (**D**) aplicada al estado actual. Antes de cada transición interna, el modelo puede generar un evento de salida, ejecutando la función de salida, (λ) que depende del estado previo a la transición. También puede haber un cambio de estado cuando hay un evento de entrada externa (δ_{ext}). La función de transición externa determina el nuevo estado basándose en el estado actual, el tiempo transcurrido en ese estado, y la entrada.

Todo modelo atómico tiene definida dos variables de estado estándar: **fase** y σ. Si no hay eventos externos, el modelo se queda en la fase durante σ, que es el tiempo que resta hasta la próxima transición interna. Si hay eventos externos, la transición externa hace que el modelo cambie de fase y σ, preparándolo para la próxima transición interna. El estado en el que se entra luego de un evento externo depende de la entrada, el estado actual, y el tiempo transcurrido en este estado, permitiendo representar el comportamiento de sistemas de tiempo continuo con eventos discretos. Por ende, la función de avance de tiempo que controla el timing de las transiciones internas simplemente devuelve el valor de σ.

Resumiendo, un estado **s** con **D**(**s**) = ∞ se dice pasivo (un estado no pasivo se dice activo). La función de transición interna δ_{int} especifica un cambio de estado en el modelo luego del tiempo dado por D. Si hay un evento externo en el instante (**t**+**e**), se ejecuta la función de transición externa δ_{ext}, y el modelo cambia inmediatamente al estado **s'**= δ_{ext}(**s**,**e**,**x**). La variable e es el tiempo transcurrido en el estado actual, y se pone en cero luego de cambio de estado.

Cada port de entrada precisa de una especificación de la transición externa, en la forma de "cuando reciba x en el port de entrada p...". La función de transición interna puede especificarse en una descripción procedural con fases y sus transiciones, con la forma "enviar y al port de salida p".

Es importante notar que no hay forma de generar una salida directamente desde un evento de entrada externa. Una salida solo puede ocurrir antes de una transición interna. Para que un evento externo provoque una salida sin demora, hay que "planificar" un estado interno con duración cero.

Los modelos básicos pueden ser acoplados en el formalismo DEVS para formar un *modelo multicomponente o acoplados*, definido por la estructura:

CM = < I, X, Y, D, {M_i}, {I_i}, {Z_{ij}}, select>

I = < P^X, P^Y> representa la definición de la interfaz modular del modelo. En este caso, $\forall$ j ∈ [1, η], $i \in$ {X, Y}, P_j^i es una definición de un port (de entrada o salida respectivamente), donde

P_j^i = { (N_j^i, T_j^i) / $\forall$ j ∈ [1, μ], (μ ∈ $\boldsymbol{N}$, μ < ∞), $N_j^i \in [i_1, i_m]$ (nombre del port), y T_j^i = Tipo del port};

X es el conjunto de eventos externos de entrada;

Y es el conjunto de eventos externos generados para salida;

D $\in$ ***N*** , D $< \infty$, es el conjunto de índices de modelos componentes; y $\forall$ i $\in$ D,

$\mathbf{M_i}$ es un modelo componente básico, definido como:

$M_i = <I_i, X_i, S_i, Y_i, \delta_{inti}, \delta_{exti}, \lambda_i, D_i>$

$\mathbf{I_i} \subseteq D$ es el conjunto de influenciados de i; y para cada $\forall$ j $\in I_i$,

$\mathbf{Z_{ij}}$: Yi $\rightarrow$ Xj es la función de traducción de salida i a j.

select: D$\rightarrow$D / $\forall$ E $\neq \{\varnothing\}$, select(E) $\in$ E es el selector ante eventos simultáneos (función de secuenciación o prioridad).

Como vemos, un modelo acoplado dice como acoplar (conectar) varios modelos componentes para formar un nuevo modelo. Este modelo puede ser empleado como componente, permitiendo construcción jerárquica. El modelo acoplado contiene, por un lado, una interfaz modular (**I**),que le permite conectarse con otros modelos básicos. Luego, se definen los conjuntos de entrada y salida del componente (**X**, **Y**), como en otros modelos básicos. Por otro lado, se define un índice (**D**) para el conjunto de componentes ($\mathbf{M_i}$) que conforman el sistema acoplado. Se determinan las influencias de cada componente ($\mathbf{I_i}$), y una especificación de acoplamiento ($\mathbf{Z_{ij}}$), en la cual la conexión de i a se especifica designando a j como el influenciado por i. Z_{ij} provee una función de traducción desde el estado de i hasta el conjunto de entradas de j. Cuando ocurre un evento interno en i, en el mismo instante envía una señal al componente j. Finalmente, la función de selección, incluye las reglas empleadas para elegir cual de los componentes inminentes (los que tienen menor tiempo al próximo evento) ejecutará su próximo evento.

Un modelo multicomponente es equivalente a un modelo básico en el formalismo DEVS, y puede ser empleado en un modelo multicomponente mayor, ya que el formalismo es cerrado bajo acoplamiento. De esta forma, un modelo puede construirse jerárquicamente a partir de submodelos DEVS. Un modelo DEVS que no se construye usando un modelo acoplado es un modelo atómico.

7.2. Cell-DEVS

El paradigma Cell-DEVS permite la descripción de modelos celulares (presentados en la sección 4) a través de su definición como modelos atómicos DEVS con distintos tipos de demoras. Cada celda será definida como un modelo atómico, y podrán utilizarse distintas clases de demoras para su comportamiento.

En los modelos Cell-DEVS con *demoras de transporte*, las celdas se definen como modelos atómicos, con el fin de acoplarlas con otras para formar un espacio de celdas completo. Se considera que los modelos son cerrados, ya que cada celda sólo puede influenciar o ser influenciada por un vecino.

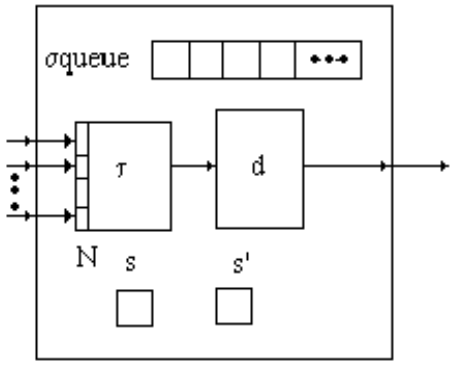


Figura 97 - Descripción informal de una celda atómica con demoras de transporte.

La **Figura 97** muestra informalmente los contenidos básicos de una celda. En este modelo atómico, una celda tiene η entradas en las que ocurren eventos externos. La celda memoriza los valores de todas las entradas, y cuando hay un nuevo evento externo, ejecuta la función booleana τ, que consume los valores de los eventos de entrada. Luego, el resultado del cálculo se demora durante **d** unidades de tiempo antes de ser transmitido a las celdas vecinas, planificando para ello un evento interno. Se debe usar una cola para mantener los valores de los resultados de los cálculos junto con su hora futura de planificación, ya que durante la demora pueden llegar nuevos eventos externos. Cuando llega la hora del evento interno, el valor se transmite por un port de salida.

Como los modelos influenciados sólo deben activarse cuando la celda influenciante cambia su estado, el resultado de la función de cálculo local sólo será transmitido si hay un cambio. Para permitir esta distinción, se guardan los valores presente y pasado de la celda.

Un modelo atómico como el descripto puede definirse formalmente como:

$$TDC = < X, Y, I, S, \theta, N, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D >$$

donde para $T < \infty \wedge T \in \{\boldsymbol{N}, \boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{\{0,1\}}\}$;

$\boldsymbol{X} \subseteq T$ es el conjunto de eventos externos de entrada;

$\boldsymbol{Y} \subseteq T$ es el conjunto de eventos externos de salida;

I = $<\eta, \mu, P^x, P^y>$ representa la definición de la interfaz modular del modelo. En este caso,

$\eta \in \boldsymbol{N}, \eta < \infty$ es el tamaño de la vecindad,

$\mu \in \boldsymbol{N}, \mu < \infty$ es la cantidad de ports de entrada/salida independientes de la vecindad, y

$\forall\ j \in [1, \eta]$, $i \in \{X, Y\}$, P_j^i es una definición de un port (de entrada o salida respectivamente),

$P_j^i = \{ (N_j^i, T_j^i) / \ \forall\ j \in [1, \eta+\mu],\ N_j^i \in [i_1, i_{\eta+\mu}]$ (nombre del port), y $T_j^i \in I_i$ (Tipo del port)},

$I_i = \{ x / x \in \boldsymbol{X}$ si $X \}$ ó $I_i = \{ x / x \in \boldsymbol{Y}$ si $i = Y \}$;

S $\subseteq T$ incluye todos los valores posibles de estados secuenciales para la celda;

θ es la definición del estado de la celda, definido como

$\theta = \{ (s, phase, \sigma queue, \sigma)$ /

$s \in S$ es el valor del estado para la celda,
phase $\in$ {activa, pasiva},
σqueue = { $((v_1,\sigma_1),...,(v_m,\sigma_m))$ / $m \in \boldsymbol{N} \wedge m < \infty) \wedge \forall (i \in \boldsymbol{N}, i \in [1,m]), v_i \in S \wedge \sigma_i \in \boldsymbol{R}_0^+ \cup \infty$}; y
$\sigma \in \boldsymbol{R}_0^+ \cup \infty$
} ;

N $\in S^{\eta}$, es el conjunto de estados de los eventos de entrada almacenados;

d $\in \boldsymbol{R}_0^+$, $d < \infty$ es la demora de transporte de la celda;

$\boldsymbol{\delta_{int}}$: $\theta \rightarrow \theta$ es la función de transición interna;

$\boldsymbol{\delta_{ext}}$: $Qx\boldsymbol{X} \rightarrow \theta$ es la función de transición externa, donde Q es el conjunto de estados definido como:

$$Q = \{ (s, e) / s \in \theta \text{ x } N \text{ x } d; e \in [0, D(s)]\};$$

$\boldsymbol{\tau}$: $N \rightarrow S$ es la función de cálculo local;

$\boldsymbol{\lambda}$: $S \rightarrow \boldsymbol{Y}$ es la función de salida; y

D: θ x N x d $\rightarrow \boldsymbol{R}_0^+ \cup \infty$, es la función de duración de vida del estado.

Cada celda tiene una interfaz bien definida, compuesta por un número fijo de ports numerados en orden ascendente. Por un lado, existe un conjunto de ports para establecer el acoplamiento interno del modelo de celdas, cada uno de los cuales estará conectado con un vecino. El número de estos ports de entrada y salida de cada celda será, por ende, igual al del tamaño de la vecindad. En el caso de precisarse otras entradas o salidas, se utilizarán los demás ports definidos.

Cada port en la interfaz tiene un nombre y un tipo. Los nombres están compuestos por un identificador (**X** para entradas; **Y** para salidas) y un número natural (número de port). Tanto los conjuntos de entrada y salida (***X*** e ***Y***) como el conjunto de estados secuenciales, así como los tipos de los ports, pueden representar cualquier conjunto finito de símbolos, pero se ha elegido trabajar con un conjunto de tipos básicos con sus respectivas álgebras: ***N***, ***Z***, ***R***, y ***Booleanos***. Esta elección se debe a que la mayoría de las aplicaciones usan datos en estos dominios, y que muchos conjuntos de símbolos pueden mapearse en estos.

El estado de la celda se define como un conjunto compuesto por:

- El valor actual de la celda;
- La fase del modelo, que puede ser activa o pasiva.
- Una cola (**σqueue**) usada para mantener los tiempos de los próximos eventos y sus valores de entrada asociados; y
- La variable $\boldsymbol{\sigma}$, que representa el tiempo simulado para el siguiente evento interno planificado;

El conjunto **N** representa el conjunto de valores de entrada de la celda, que tiene la forma de una η-upla $(s_1,..., s_\eta)$ donde $s_i \in S$. Este conjunto registra los valores actuales usados para calcular el valor futuro a través de la función de cálculo local $\boldsymbol{\tau}$.

La función de duración de vida **D** se usa para controlar el tiempo de duración del estado de una celda. En este caso, **D**(s, phase, σqueue, σ, N, d) = t representa el tiempo durante el cual, si no hay eventos externos, el modelo atómico conservará el estado actual. Tanto la función de vida del estado como la demorade la celda tienen dominio en los números reales, ya que la base de tiempo seleccionada es continua.

Los objetivos de la ejecución de las funciones de transición (δ_{int}, δ_{ext}) y de la de salida (λ) son similares a los definidos para otros modelos DEVS. La función de transición interna se usa para definir cambios de estado debido a eventos internos, y la función de transición externa expresará la ocurrencia de eventos externos.

En cambio, la semántica de estas funciones será diferente a la de otros modelos DEVS. Esto se debe a que cada celda puede tener asociada una demora de transporte (**d**), que permite postergar la ejecución de la función de transición interna. La construcción de demora de transporte permite que el cambio de estado ante la ocurrencia de un evento externo sea demorado, y que el estado del sistema sólo cambie al consumirse el tiempo correspondiente a la demora. Otra diferencia es que una vez ejecutada la función de transición interna, los modelos DEVS atómicos pasan a un estado pasivo hasta recibir nuevos eventos externos. En cambio, la introducción de demoras de transporte implica que una celda debe quedar activa durante la duración de la demora, ya que en ese lapso se pueden recibir nuevos eventos externos. Esto puede provocar que haya varios eventos internos planificados a futuro.

La semántica para la función de transición externa es la siguiente: la llegada de un evento externo indica que un vecino ha cambiado. Por ende, la celda debe activarse, y calcular su función de cómputo local. La demora de transporte indica que se planificará la ejecución futura de la función de transición interna, para lo cual se almacenan los valores de la demora y de la entrada en la cola **σqueue**. Asimismo, si al hacer el cálculo el estado de la celda no cambia, sus vecinos tampoco pueden cambiar. Por ende, el estado actual no se encola para ser transmitido.

Una celda se pone en estado pasivo sólo cuando no tiene más eventos planificados, o sea, cuando la cola **σqueue** esté vacía. El valor de la variable **σ** será, por ende, igual al primer valor de esta cola. Si la celda está activa, la llegada de un nuevo evento implica que, además de encolar el elemento, los valores de σ almacenados en la cola debe ser actualizados para reflejar el tiempo transcurrido (e).

La función de transición interna es función de la demora de la celda. Esta función (y la de salida) deben activarse cuando σ=0. Como la demora de transporte ha expirado, debe transmitirse el nuevo estado de la celda. La función de salida **λ** se ejecuta antes de la función δ_{int}, como en cualquier otra especificación DEVS, la que al activarse tomará el primer elemento de la cola de espera para ser transmitido. La función de transición interna sólo elimina el primer miembro de la cola (cuyo valor ya fue transmitido), y actualiza los tiempos de la cola de espera y de la variable de estado σ.

El comportamiento detallado de estas funciones puede verse en la siguiente figura.

```
δext((s, phase, σqueue, σ, N, d), e, x) = (s, phase, σqueue, σ, N, d)
{
 /* Hay un evento de entrada: calcular τ, y si el nuevo valor es igual al anterior, no hacer nada */
     Actualizar los valores de N con el valor de x;
     s' = τ(N);          /* Calcula la función de cálculo local */

     if (s' != s) then       /* Los valores deben transmitirse sólo si cambia el estado */
          s = s';
          if (phase = pasiva) then /* La celda está pasiva. Activarla. */
                    phase = activa;
                    σ = d;                    /* Demora de transporte */
          else
                     if (d < σ) σ = d endif;  /* Las demoras de una celda pueden ser variables, y
                     podría llegar un evento externo con menor demora que las ya almacenadas */
                    for (ai ∈ σqueue) ai.σ = ai.σ - e;  /* La celda está activa. El tiempo transcurrido
                    σ = σ - e;                  debe actualizar los valores futuros de σqueue */
          endif
          σqueue = insertar(σqueue, <s, d>);       /* Inserción ordenada por σ */
     }
}

δint(s, phase, σqueue, σ) = (s, phase, σqueue, σ)
{         /* Los tiempos de σqueue deben actualizarse */
          for (ai ∈ σqueue) ai.σ = ai.σ - first(σqueue.σ);
          σqueue = tail (σqueue);

          if (empty(σqueue)) then /* No hay más eventos planificados */
                    phase = pasiva;
                    σ = ∞;
          else              /* planificar el siguiente evento */
                    phase = activa;
                    σ = first(σqueue.σ);
          endif
}

λ(s) {
  return first(σqueue.value);
}
```

Figura 98 - Definición de δ_{int}, δ_{ext} y λ para modelos TDC.

Otro tipo de demora interesante para las celdas de los modelos celulares es la *demora inercial*. Esta construcción permite representar ciertos fenómenos cuya semántica incluye comportamiento con desalojo, permitiendo descartar entradas a las celdas si su valor no se mantiene durante un período determinado. Por otro lado, si el flujo de entrada es constante durante ese tiempo (llamado demora inercial) el estado debe cambiar.

La **Figura 99** ejemplifica el comportamiento de una demora inercial para una celda atómica.

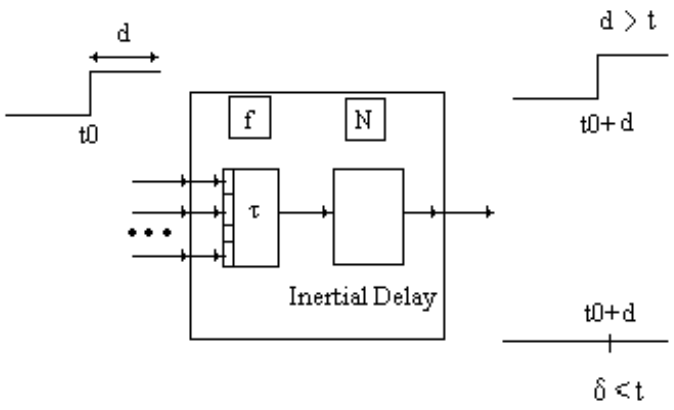


Figura 99 - Comportamiento de un modelo atómico de celdas con demora inercial.

Un modelo atómico de celda con demoras inerciales puede especificarse como:

$$IDC = < X, Y, I, S, \theta, N, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D >$$

donde para $T < \infty \wedge T \in \{\boldsymbol{N}, \boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{\{0,1\}}\}$;

$\boldsymbol{X} \subseteq T$ es el conjunto de eventos externos de entrada;

$\boldsymbol{Y} \subseteq T$ es el conjunto de eventos externos de salida;

I = $<\eta, \mu, P^x, P^y>$ representa la definición de la interfaz modular del modelo. En este caso,
$\eta \in \boldsymbol{N}, \eta < \infty$ es el tamaño de la vecindad,
$\mu \in \boldsymbol{N}, \mu < \infty$ es la cantidad de ports de entrada/salida independientes de la vecindad, y
$\forall\ j \in [1, \eta]$, $i \in \{X, Y\}$, P_j^i es una definición de un port (de entrada o salida respectivamente),
$P_j^i = \{ (N_j^i, T_j^i) / \ \forall\ j \in [1, \eta+\mu],\ N_j^i \in [i_1, i_{\eta+\mu}]$ (nombre del port), y $T_j^i \in I_i$ (Tipo del port)$\}$, $I_i = \{ x / x \in \boldsymbol{X}$ si $i = X \}$ ó $I_i = \{ x / x \in \boldsymbol{Y}$ si $i = Y \}$;

S $\subseteq T$ incluye todos los valores posibles de estados secuenciales para la celda;

θ es la definición del estado de la celda, definido como

$\theta = \{ (s, phase, f, \sigma) / s \in S$, phase $\in$ {activa, pasiva}, $f \in T$, y $\sigma \in \boldsymbol{R}_0^+ \cup \infty \}$;

N $\in S^\eta$, es el conjunto de estados de los eventos de entrada almacenados;

d $\in \boldsymbol{R}_0^+$, $d < \infty$ es la demora inercial de la celda;

$\boldsymbol{\delta_{int}}$: $S \rightarrow S$ es la función de transición interna;

$\boldsymbol{\delta_{ext}}$: $Q \times X \rightarrow \theta$ es la función de transición externa, donde Q es el conjunto de estados definido como:

$$Q = \{ (s, e) / s \in \theta \times N \times d \ \wedge e \in [0, D(s)]\};$$

τ: N $\rightarrow$ S es la función de cálculo local;

λ: S $\rightarrow Y$ es la función de salida; y

D: θ x N x d $\rightarrow \boldsymbol{R}_0^+ \cup \infty$, es la función de duración de vida del estado.

Como puede verse, la mayoría de los conjuntos y funciones fueron definidos de forma similar a los presentados para modelos con demoras de transporte, pero existen algunas diferencias. Por un lado, la definición para el estado de una celda ha sido modificada. En este caso, s, phase y σ tienen el mismo significado definido anteriormente, pero f representa el valor futuro factible para la celda. Si el valor de entrada de la celda se mantiene durante la demora inercial, f se convertirá en el estado actual de la celda.

La definición de la variable **d** corresponde ahora a una demora inercial, lo que motiva un cambio en la semántica de las funciones de transición, que ahora deben modelar demoras inerciales. Para este caso, el comportamiento de las funciones de transición se ha definido como sigue:

```
δext((s, phase, f, N, d, σ), e, x) = (s, phase, f, N, d, σ)
{
        Actualizar los valores de los eventos de entrada con el valor de x;
        s' = τ(N);

        if (s != s') then
          s= s';            /* El nuevo estado es el recién calculado */

            if (phase = pasiva) then
                phase = activa;
                σ = d;    /* Demora inercial */
            else
                σ = σ - e;
                if (σ > 0 .AND. f != s)  σ = d;  /* Remoción: el estado calculado es distinto al
futuro */
            endif
        endif
        f = s;    /* El estado futuro es el recién obtenido */
}

δint(s, phase, f, σ) = (s, phase, f, σ)
{
        if (σ = 0) then
                phase = pasiva;
                σ = ∞;
        endif
}

λ(s) {
        return s;
}
```

Figura 100 - Funciones de transición interna y externa para modelos con demoras inerciales.

Podemos ver que el objetivo de la función de transición externa es almacenar el valor actual para la celda, y detectar si la entrada se conserva durante la demora inercial. Si no es así, la entrada anterior es removida. Al llegar la hora planificada por la demora, se transmite el valor actual de la celda.

Los modelos atómicos de celdas con distintas demoras pueden acoplarse con otros modelos para formar un *modelo Cell-DEVS multicomponente o acoplado*. Éstos son definidos como un espacio consistente de celdas atómicas conectadas por una relación de vecindad. Cada uno de los componentes es una celda como las definidas anteriormente. En esta especificación se consideran modelos cerrados (es decir, que no pueden acoplarse con otros modelos de base). Por ende, al no usarse entradas ni salidas, tampoco hay necesidad de definir una interfaz para el modelo.

Un modelo ejecutable de celdas Cell-DEVS acopladas puede definirse como:

$$CC = < n, \{t_1,..., t_n\}, \eta, N, C, B, Z, select >$$

donde

n es la dimensión del espacio de celdas;

$\{\mathbf{t_1,...,t_n}\}$ es la cantidad de celdas en cada una de las dimensiones;

$\boldsymbol{\eta}$ es el tamaño de la vecindad;

N es el conjunto de vecindad;

C es el espacio de celdas;

B es el conjunto de celdas del borde;

Z es la función de traducción; y

select es la función de selección ante eventos simultáneos.

En este caso,

$\mathbf{n} \in \boldsymbol{N}$;

$\{\mathbf{t_1,...,t_n}\} \in \boldsymbol{N}$, con $t_k < \infty \; \forall \; k \in [1,n]$;

$\boldsymbol{\eta} \in \boldsymbol{N}$;

$\mathbf{N} = \{ (v_{k1},...,v_{kn}) / \forall (k \in \boldsymbol{N}, k \in [1, \eta] \wedge i \in \boldsymbol{N}, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \wedge (t_i - | v_{ki} | \geq 0) \}$;

$\mathbf{C} = \{ C_C / c \in \mathbf{I} \wedge C_C = < I_C, X_C, Y_C, S_C, N_C, d_C, \delta_{intC}, \delta ext_C, \tau_C, \lambda_C, D_C >$ es un componente TDC ó IDC $\}$, siendo

$$\mathbf{I = \{ (i_1,...,i_n) / (i_k \in N \wedge i_k \in [1, t_k]) \; \forall \; k \in [1, n] \} \; (1)}$$

B = {∅} si el espacio de celdas es toroidal; o

B = {C_b / $C_b \in$ C con b $\in$ **I** }, con **I** definido como en (1). En este caso, el conjunto B está sujeto a la restricción que $\tau_b \neq \tau_c = \tau \ \forall (C_c \notin B) \wedge (C_b \in B)$.

B = {C_b / $C_b \in$ C con b $\in$ L}, siendo L = { $(i_1,...,i_n)$ / $i_j = 0 \vee i_j = t_j \ \forall j \in [1, n]$ }, y sujeto a que $\tau_b \neq \tau_c = \tau \ \forall c \notin L$.

Definición

Si llamamos c a la posición de una celda, donde c = $(i_1,...,i_n)$; y sea V $\in$ $\boldsymbol{Z}^n$ la n-upla definida como V = $(v_1, ..., v_n)$, con $v_i \in \boldsymbol{Z}$, $| v_i | \leq t_i$ entonces se define como

$$D = C +_t V \quad (2)$$

a la n-upla D = $(j_1,...,j_n)$ definida por: $j_k = (i_k + v_k) \bmod (t_k) \ \forall k \in [1, n]$.

Notación 11

$\forall k \in \boldsymbol{N}$, k $\in$ [1, η], se denotará V_k = { $(v_{k1},...,v_{kn})$ / $(v_{k1},...,v_{kn}) \in$ N } al k-ésimo elemento de una lista de vecindad.

Notación 12

Se denotará P_c^{iq} al port $P_q^i \in I_c$, con *i* $\in$ {X, Y}, y (q $\in$ ***N***, q $\in$ [1, η]), donde $I_c \subseteq C_c$, y C_c es un componente TDC ó IDC

Z: $I_C \rightarrow I_D$ es la función de traducción, definida (usando (5)) como:

$Z(P_C{}^{Y_q}) = P_D{}^{X_q}$, $\forall$ (q $\in$ ***N***, q $\in$ [1,η]), donde $\forall V_k \in$ N, $D = C +_t V_k$; y
$Z(P_C{}^{X_q}) = P_D{}^{Y_q}$, $\forall$ (q $\in$ ***N***, q $\in$ [1,η]), donde $\forall V_k \in$ N, $D = C -_t V_k$.

select = { $(s_1, ..., s_n)$ / $s_i \in \boldsymbol{N}$, $\forall i \in [0, n]$ }, con la restricción que select $\subseteq t_1xt_2x...xt_n \rightarrow t_1xt_2x...xt_n$.

En este caso, el espacio de celdas **C** es un modelo acoplado definido como un arreglo de modelos atómicos Cell-DEVS de tamaño fijo (t_1 x ... x t_n).

Cada celda tiene un conjunto de **η** vecinas, definidas por el conjunto de vecindad (**N**). Este conjunto está representado como una lista de tuplas de dimensión n que definen la posición relativa entre la celda origen y sus vecinos. Estos índices no pueden exceder el límite del espacio de celdas.

El conjunto **B** define el borde del espacio de celdas, y puede ser de dos tipos distintos. Si B = {∅}, toda celda en el espacio tendrá el mismo comportamiento: las celdas en un borde se conectan con las que están el borde opuesto usando la relación de vecindad inversa. En cambio,

si el conjunto de borde es no vacío, las celdas tendrán un comportamiento diferente a las otras del modelo (por ejemplo, pueden generar su estado, o consumir los estados de los vecinos).

La función **Z** permite definir el acoplamiento entre celdas en el modelo. Esta función traduce las salidas del q-ésimo port de salida en la celda C_c en valores para el q-ésimo port de entrada de la celda C_D (y viceversa). Cada port de salida corresponderá a un vecino, y cada port de entrada estará asociado con una celda en la vecindad inversa [Zei76].

Los nombres de los ports se generan usando la siguiente notación: $P_C{}^{X_q}$ se refiere al q-ésimo port de entrada de la celda C_c, y $P_C{}^{Y_q}$ al q-ésimo port de salida. Estos ports corresponden con los nombres de ports denotados como X_q o Y_q para cada celda. El número de la celda con la cual debe acoplarse el modelo será generado sumando los números de la lista de vecinos al número de la celda actual. El primer port de salida de una celda estará conectado con el primer port de entrada del vecino, de acuerdo con el orden de la lista de vecinos.

La definición presentada sólo incluye permite vecindades regulares y vecinos en celdas adyacentes. Si fueran necesarios otros tipos de vecindades (incluyendo distintas vecindades para cada celda), la definición puede extenderse. Aquí,

$$N = \{N_C / c \in \mathbf{I}\} \text{ con } \mathbf{I} \text{ definido como en (4)}$$

donde

$$N_C = \{ (v_{k1},...,v_{kn})_C / \forall (k \in \boldsymbol{N}, k \in [1, \eta_C]) \wedge (i \in \boldsymbol{N}, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \wedge (ti - | v_{ki} | \geq 0) \};$$

Por último se extiende la definición de los modelos Cell-DEVS acoplados cerrados a *modelos Cell-DEVS acoplados genéricos*. La idea es extender la definición de los primeros para permitir la especificación de espacios Cell-DEVS que puedan ser combinados con otros modelos básicos en una jerarquía DEVS.

Para permitir la definición múltiple de submodelos y su acoplamiento, un modelo acoplado Cell-DEVS genérico puede representarse como:

$$GCC = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z, select >$$

En este caso,

Ylist es la lista de acoplamiento de salida;

Xlist es la lista de acoplamiento de entrada;

I representa la definición de la interfaz modular del modelo;

X es el conjunto de eventos externos de entrada;

Y es el conjunto de eventos externos de salida;

n es la dimensión del espacio de celdas;

$\{\mathbf{t_1,...,t_n}\}$ es la cantidad de celdas en cada una de las dimensiones;

η es el tamaño de la vecindad;

N es el conjunto de vecindad;

C es el espacio de celdas;

B es el conjunto de celdas del borde;

Z es la función de traducción; y

select es la función de selección ante eventos simultáneos.

Donde, para $T < \infty \wedge T \in \{\boldsymbol{N}, \boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{\{0,1\}}\}$;

Ylist $\subseteq \{ (i_1,...,i_n) / i_k \in [1,t_k] \forall k \in [1,n], k \in \boldsymbol{N}\}$;

Xlist $\subseteq \{ (i_1,...,i_n) / i_k \in [1,t_k] \forall k \in [1,n], k \in \boldsymbol{N}\}$;

I = $< P^X, P^Y >$ representa la definición de la interfaz modular del modelo. En este caso,
$\forall$ c $\in$ **I**, con **I** definido como en (4) $i \in \{X, Y\}$, P_c^i es una definición de un port,
$P_c^i = \{ (N_c^i, T_c^i) / \forall c \in i\text{list}, N_c^i \in i(c)$ (nombre del port), y $T_c^i \in T$ (Tipo del port)};

$\boldsymbol{X} \subseteq T$;

$\boldsymbol{Y} \subseteq T$;

n $\in \boldsymbol{N}$;

$\{\mathbf{t_1,...,t_n}\} \in \boldsymbol{N}$, con $t_k < \infty \forall k \in [1,n]$;

$\boldsymbol{\eta} \in \boldsymbol{N}$;

N = $\{ (v_{k1},...,v_{kn}) / \forall (k \in \boldsymbol{N}, k \in [1, \eta]) \wedge (i \in \boldsymbol{N}, i \in [1, n]); v_{ki} \in \boldsymbol{Z} \wedge (ti - | v_{ki} | \geq 0) \}$;

C = $\{ C_c / c \in \mathbf{I} \wedge C_c = < I_c, X_c, Y_c, S_c, N_c, d_c, \delta_{int c}, \delta ext_c, \tau_c, \lambda_c, D_c >$ es un componente TDC ó IDC tal como los definidos anteriormente }, siendo **I** definido como en (4);

B = $\{\varnothing\}$ si el espacio de celdas es toroidal; o
B = $\{C_b / C_b \in C$ con $b \in \mathbf{I}\}$, con **I** definido como en (4). En este caso, el conjunto B está sujeto a la restricción que $\tau_b \neq \tau_c = \tau \ \forall (C_c \notin B) \wedge (C_b \in B)$.

$B = \{C_b / C_b \in C$ con $b \in L\}$, siendo $L = \{ (i_1,...,i_n) / i_j = 0 \vee i_j = t_j \forall j \in [1, n] \}$, y sujeto a que $\tau_b \neq \tau_c = \tau \forall c \notin L$.

Z: $I_C \rightarrow I_D$ es la función de traducción, definida (usando (3)) como:

$Z(P_c^{Y_q}) = P_D^{X_q}, \forall (q \in \boldsymbol{N}, q \in [1,\eta])$, donde $\forall V_k \in N$, $D = C +_t V_k$; y
$Z(P_c^{X_q}) = P_D^{Y_q}, \forall (q \in \boldsymbol{N}, q \in [1,\eta])$, donde $\forall V_k \in N$, $D = C -_t V_k$.

select = { (s_1, ..., s_n) / $s_i \in \boldsymbol{N}$, $\forall$ i $\in$ [0, n] }, con la restricción que select $\subseteq t_1xt_2x...xt_n \rightarrow t_1xt_2x...xt_n$.

Se pueden encontrar algunas diferencias con la especificación de los espacios Cell-DEVS cerrados. Primero, como en todo modelo acoplado DEVS que admite entradas y salidas, han sido incluidos los conjuntos ***X*** e ***Y***. Como estos modelos pueden ser acoplados con otros modelos DEVS, también ha sido definida la interfaz **I**.

La función **Z** se sigue usando para llevar a cabo el acoplamiento interno del espacio de celdas. Finalmente, se han incluido dos nuevos conjuntos: **Xlist**, una lista de las posiciones de las celdas donde se reciben eventos externos al modelo acoplado; e **Ylist**, una lista de posiciones de celdas cuyas salidas serán recolectadas para ser enviadas a otros modelos en la jerarquía.

En modelos DEVS acoplados, la ocurrencia de eventos internos en un modelo es precedida por la ejecución de la función de salida λ. Esta función transmite el valor a otros modelos a través de ports en la interfaz. Para modelos Cell-DEVS, el comportamiento será distinto. Primero, si ocurre un evento en una celda, sus vecinos serán influenciados automáticamente a través de la ejecución de la función Z. Por otro lado, ciertas celdas en el espacio serán elegidas como celdas de entrada y salida, y serán incluidas en las listas Xlist e Ylist respectivamente. Los valores de estas celdas serán considerados como las entradas y salidas de todo el espacio de celdas.

La función select se define como una lista de posiciones en la vecindad. La lista se ordena de acuerdo con el criterio de selección a ser utilizado cuando más de una celda está activa simultáneamente.