Desarrollo de modelos de tráfico vehicular en GALATEA

Carlos Daniel Pérez R.
Tutor Prof. Kay A. Tucci Kelerer

22 - Junio 2010

Objetivo General

- Desarrollar e implementar modelos para tráfico vehicular unidireccional utilizando tres variantes del enfoque microscópico, es decir, el modelo del conductor inteligente, modelo de mapas acoplados y modelo de autómatas celulares.

Objetivos Específicos

- Desarrollar tres modelos de tráfico vehicular unidireccional, con ecuaciones diferenciales ordinarias, mapas acoplados y autómatas celulares.
- Implementar los tres modelos en GALATEA.
- Realizar algunas simulaciones de cada modelo en tres escenarios diferentes: Un canal unidireccional, un canal unidireccional con semáforo y un canal unidireccional con incorporación de vehículos.

Algunos antecedentes

Años treinta dinámica de fluidos Greenshields

Años cincuenta macroscópica y microscópica

Años Noventa Información estadística y poder de computo

Relación Modelos - Escalas de tiempo

Modelos microscópicos			
Caso	Modelo	Escalas de tiempo	Aspecto
Dinámica de vehículo	Sub-microscópico	0.1 segundo 1 segundo	Cola-carros, frenado reacción y brecha tiempo
Dinámica de tráfico	Carro siguiente	10 segundos	Aceleración y frenado
Modelos macroscópicos			
Caso	Modelo	Escalas de tiempo	Aspecto
Dinámica de tráfico	Dinámica de fluidos	1 minuto 10 minutos 1 hora	período de luz de tráfico ondas de arranque-frenado hora pico
	Asignación de tráfico	1 día	Comportamiento humano día-día
Planificación de transporte	Demanda de tráfico Estadística Pronóstico	1 Año 5 Años 50 Años	Medidas de construcción Cambios-Estructura espacial Cambios-Demográficos

simulación enfocada en modelos microscópicos

Autómatas celulares

Autómatas celulares

Nagel-Schreckenberg

Estado actual	111	110	101	100	011	010	001	000
Nuevo estado	1	0	1	1	0	0	0	0

Mapas acoplados

Iteración de mapas acoplados

Newell

$$x_i(t + \Delta t) = v_i(t)\Delta t + x_i(t) \ . \tag{1}$$

$$v_i(t + \Delta t) = G_i(s_i(t), v_i(t)) \ , \tag{2}$$

Conductor inteligente

Conductor inteligente

Helbing, Hennecke y Treiber

$$\begin{cases} \dfrac{dv_i}{dt} = a_i \left[1 - \left(\dfrac{v_i}{v_i^*} \right)^{\delta} - \left(\dfrac{s_i^*}{s_i} \right)^2 \right] \\ \dfrac{dx_i}{dt} = v_i \end{cases} , \tag{3}$$

$$s_i^* = s_0 + v_i T + \frac{v_i (v_i - v_{i-1})}{2\sqrt{a_i b_i}} , \tag{4}$$

$$s_i = x_{i-1} - x_i . \tag{5}$$

Algunos conceptos importante

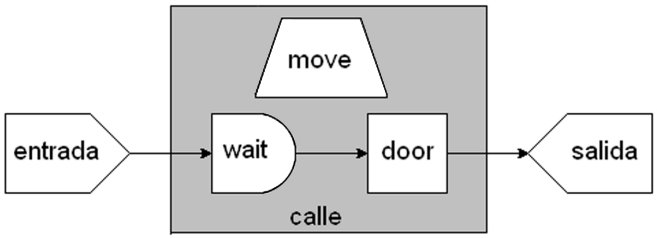
GALATEA — lenguajes, compiladores, interpretadores y colección de bibliotecas

- Nodos — sub-sistemas
  - **I**nput — mensajes
  - **R**esource — sim. recursos
  - **E**xit — eliminación
  - **A**utonomous — progr. eventos
- gSpace — recrea el espacio

Detalles de los modelos base

- Una vez que los vehículos se incorporan en la vía conservaran el orden en que entran a la carretera hasta el momento de la salida, el vehículo i-esimo siempre será el siguiente del $(i-1)$-esimo.
- Los vehículos ocupan un área determinada y, por tanto, una posición en el espacio sólo puede ser ocupada por un vehículo.
- Los vehículos deben tener aceleración, velocidad, y capacidad de frenado finita.
- En los modelos no habrá estudio de colisiones ni se hará simulaciones que las incluyan.
- La carretera es unidireccional, por lo que todos los vehículos se movilizan de izquierda a derecha.
- Los vehículos guardan una distancia mínima s_i^* entre sí que no dependerá directamente del modelo implementando.
- La posición, velocidad y aceleración de cada uno de los vehículos son individuales, es decir, el vehículo i tendrá posición x_i, velocidad v_i y aceleración a_i.

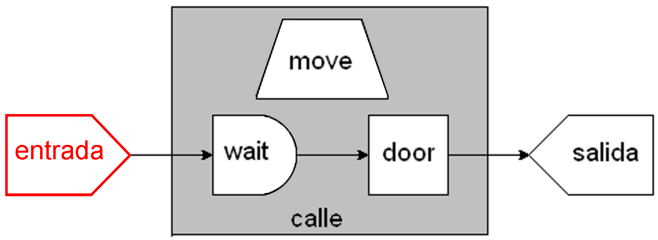
Modelo Base - Via



Esquema del modelo base en Galatea.

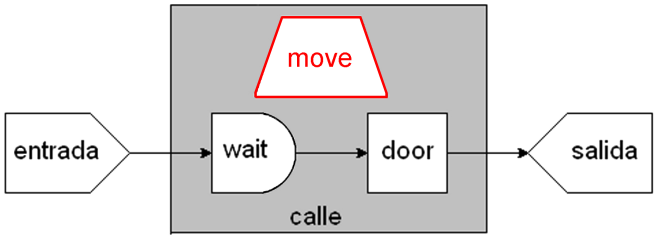
```
import galatea.glider.*;
import galatea.gspaces.*;
public class Via {
  static double l;
  static Entrada entrada = new Entrada();
  static Space calle = new Space("calle",0,6.0,6.0,1.0,1.0,"Regla")
  static Salida salida = new Salida();
  public static void main(String[] args) {
      setExperiment(args);
      calle.addWall(0.0, 0.0, l, 0.0);
      calle.addWall(0.0, 12.0, l, 12.0);
      calle.addWall(0.0, 0.0, 0.0, 12.0);
      calle.addWall(l, 0.0, l, 12.0);
      calle.addDoor(l, 0.0, l, 12.0, salida, 'U', 1, 0.0, 0.0);
      calle.build();
      Glider.trace("Via.trc");
      Glider.stat("Via.sta");
      Glider.setTsim(10000);
      Glider.act(entrada, 0);
      Glider.act(calle.getMove(), 1);
      Glider.process();
  }
  static void setExperiment(String[] args) {
      args = Galatea.checkArgs(args);
      l = Galatea.doubleArg(args, "Longitud", "01000.0");
       Glider.addPath(Galatea.getExperDir());
  }
}
```

Entrada



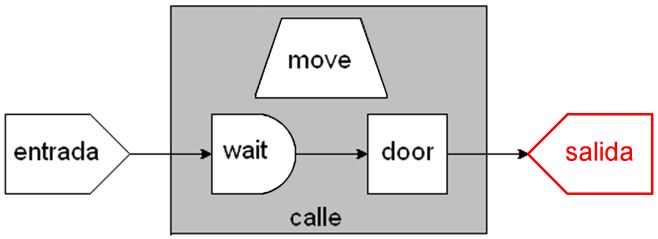

```
import galatea.glider.*;
import galatea.gspaces.*;
class Entrada extends Node {
    Entrada() {
//      ----------------------
        super("entrada", 'I');
//      ----------------------
        Glider.nodesl.add(this);
    }
    public void fact() {
//      --------------------
        it(GRnd.unif(1, 4));
//      --------------------
        Glider.mess.addField("x", 0.1);
        Glider.mess.addField("y", Via.calle.getYIn());
        Glider.mess.addField("v0", 12);
        Glider.mess.addField("tU",Glider.mess.getNumber()%2);
        Glider.mess.addField("p", null);
        Glider.mess.addField("c", 0);
        Glider.mess.addField("s", 0);
        Glider.mess.addField("d", null);
        Glider.mess.addField("cut", null);
        Glider.mess.addField("vvx", 0);
        Glider.mess.addField("vy", 0);
        /* ************************************** */
        /* ACA VA EL CODIGO PARTICULAR DE CADA MODELO */
        /* ************************************** */
        GSpace.sendto(Glider.mess, this, Via.calle);
    }
}
```

Regla




```
import galatea.glider.*;
import galatea.gspaces.*;
public class Regla {
    public Regla(){}
    public double[] move(Move e, Message m, Cell c) {
        double[] nCoord = new double[4];
        /* ************************************** */
        /* ACA VA EL CODIGO PARTICULAR DE CADA MODELO */
        /* ************************************** */
        nCoord = e.dChecking(m, c, nCoord, e.getDoor(1));
       return nCoord;
    }
}
```

Salida




```
import galatea.glider.*;
class Salida extends Node {
    Salida() {
//      ---------------------
        super("salida", 'E');
//      ---------------------
        Glider.nodesl.add(this);
    }
    public void fscan() {
        /* ****************************************** */
        /* ACA VA EL CODIGO PARTICULAR DE CADA MODELO */
        /* ****************************************** */
    }
}
```

Regla y Método

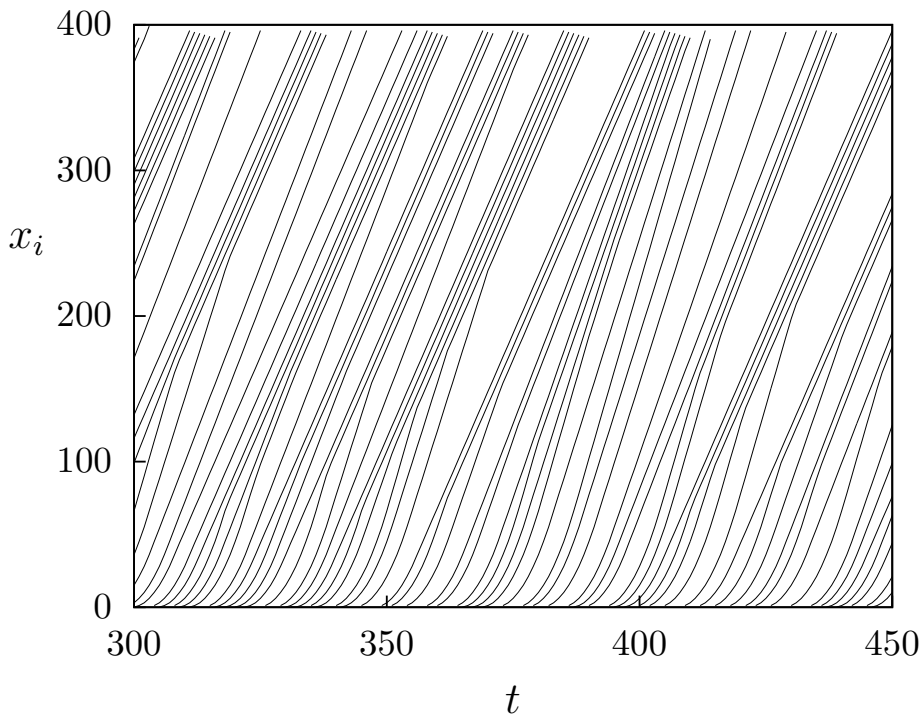
$$x_i(t + it) = x_i(t) + v_i(t + it) \,, \qquad (6)$$

```
if(lastUpdate < Glider.getTime()){
    lastUpdate=Glider.getTime();
    desplazamiento();
}
double[] nCoord = new double[4];
nCoord[0] = m.getdoubleValue("x")+ m.getdoubleValue("vvx");
nCoord[1] = m.getdoubleValue("y");
```

```
void desplazamiento(){
    Cell[] c = Via.calle.getMove().getCells();
    Message m;
    int i;
//  -------
    int v0;
//  -------
    int vx;
    int vx1;
    for(i=0;i<c.length;i++){
        if(!c[i].isEmpty()){
            m = (Message)(c[i].getAg().getDat(1));
            v0 =m.getintValue("v0");
            vx = m.getintValue("vvx");
            vx1 = 0;
            while(vx1 < v0 && vx1 < vx + 1){
                if(c[i + vx1 + 1].isEmpty())
                    vx1++;
                else
                    break;
            }
//          ----------------------
            m.setField("vvx", vx1);
//          ----------------------
        }
    }
}
```

Entrada

```
import galatea.glider.*;
import galatea.gspaces.*;
class Entrada extends Node {
    Entrada() {
        super("entrada", 'I');
        Glider.nodesl.add(this);
    }
    public void fact() {
//      --------------------
        it(GRnd.unif(1, 4));
//      --------------------
        Glider.mess.addField("x", 0.1);
        Glider.mess.addField("y", Via.calle.getYIn());
//      ------------------------------------------------
        Glider.mess.addField("v0",GRnd.unif(8.0, 12.0));
//      ------------------------------------------------
        Glider.mess.addField("tU",Glider.mess.getNumber()%2);
        Glider.mess.addField("p", null);
        Glider.mess.addField("c", 0);
        Glider.mess.addField("s", 0);
        Glider.mess.addField("d", null);
        Glider.mess.addField("cut", null);
        Glider.mess.addField("vvx", 0);
        Glider.mess.addField("vy", 0);
        GSpace.sendto(Glider.mess, this, Via.calle);
    }
}
```



Posición vs. tiempo.

Regla

$$x_i(t + it) = x_i(t) + v_i(t)\ \ it + \frac{a_i(t + it)\ \ it^2}{2}\ , \qquad (7)$$

```
if(lastUpdate < Glider.getTime()){
    lastUpdate=Glider.getTime();
    desplazamiento();
}
//-------------------------------------------------------
nCoord[0] = m.getdoubleValue("x")
+ m.getdoubleValue("vvx") * e.getIt()
+ (1/2) * m.getdoubleValue("a")* Math.pow(e.getIt(),2);
//-------------------------------------------------------
nCoord[1]=m.getdoubleValue("y");
```

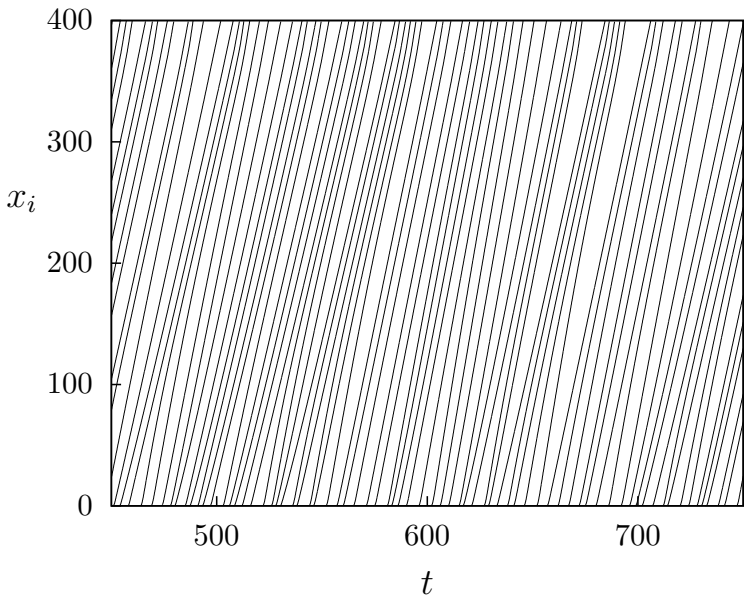
$$s_i(t) = x_{i-1}(t) - x_i(t)\ . \qquad (8)$$

$$s_i^*(t) = s_0 + v_i\, T + \frac{v_i(v_i - v_{i-1})}{2\sqrt{a_i^* b}}\ , \qquad (9)$$

$$a_i(t + it) = a_i^* \left[1 - \left(\frac{v_i(t)}{v_i^*}\right)^2 - \left(\frac{s_i^*(t)}{s_i(t)}\right)^2\right]\ , \qquad (10)$$

$$v_i(t + it) = a_i(t + it)\ \ it + v_i(t) \qquad (11)$$

```
void desplazamiento(){
    double t = Via.calle.getIt();
    for(int n=1; n<=Via.calle.getWait().getEl().ll();n++){
      Message m = (Message)Via.calle.getWait().getEl().getDat(n);
      Message ant = (Message)m.getValue("anterior");
      double v_m = m.getdoubleValue("vvx");
                    ...
      if(ant!=null){
        double x_ant = ant.getdoubleValue("x");
        double v_ant = ant.getdoubleValue("vvx");
//      -----------------------
        double g = x_ant - x_m;
//      -----------------------
        if(g>stop){
//        ------------------------------------------------------------
          double gg =g0+v_m*T+v_m*((v_m-v_ant)/(2*Math.sqrt(a0*b)))
          a = a0*(1-Math.pow((v_m/v0),2)-(Math.pow((gg/g),2)));
//        ------------------------------------------------------------
        }else{
          if(ant.getdoubleValue("vvx")!=0){
              a = 0;
          }else{
              a = 0;
              v_m = 0;
          }
        }
      } else {
//      --------------------------------
        a = a0*(1-Math.pow((v_m/v0),2));
//      --------------------------------
      }
      if(a >= 0)
//      --------------------
        m.setField("a", a);
      vx = a*t + v_m;
      m.setField("vvx", vx);
//    ----------------------
    }
```



Posición vs. tiempo

```
import galatea.glider.*;
import galatea.gspaces.*;
class Entrada extends Node {
    Message ant = null;
    Entrada() {
        super("entrada", 'I');
        Glider.nodesl.add(this);
    }
    public void fact() {
        it(GRnd.unif(2, 6));
        Glider.mess.addField("x", 0.1);
        Glider.mess.addField("y", Via.calle.getYIn());
//      ------------------------------------------------
        Glider.mess.addField("v0", GRnd.unif(8.0, 12.0));
//      ------------------------------------------------
        Glider.mess.addField("tU",Glider.mess.getNumber()%2);
        Glider.mess.addField("p", null);
        Glider.mess.addField("c", 0);
        Glider.mess.addField("s", 0);
        Glider.mess.addField("d", null);
        Glider.mess.addField("cut", null);
        Glider.mess.addField("vvx", 0.0);
        Glider.mess.addField("vy", 0.0);
//      ----------------------------
        Glider.mess.addField("a", 0.0);
        Glider.mess.addField("b",0.9);
        Glider.mess.addField("a0",11.1);
        Glider.mess.addField("g0",4.0);
        Glider.mess.addField("T",1.5);
        Glider.mess.addField("anterior",ant);
        if(ant!=null)
            ant.addField("posterior", Glider.mess);
        ant = Glider.mess;
        GSpace.sendto(Glider.mess, this, Via.calle);
    }
}
```

Entrada

$$s_i = x_{i-1} - x_i \ . \tag{12}$$

$$s_i^* = s_0 + v_i T + \frac{v_i(v_i - v_{i-1})}{2\sqrt{a_i b_i}} \ , \tag{13}$$

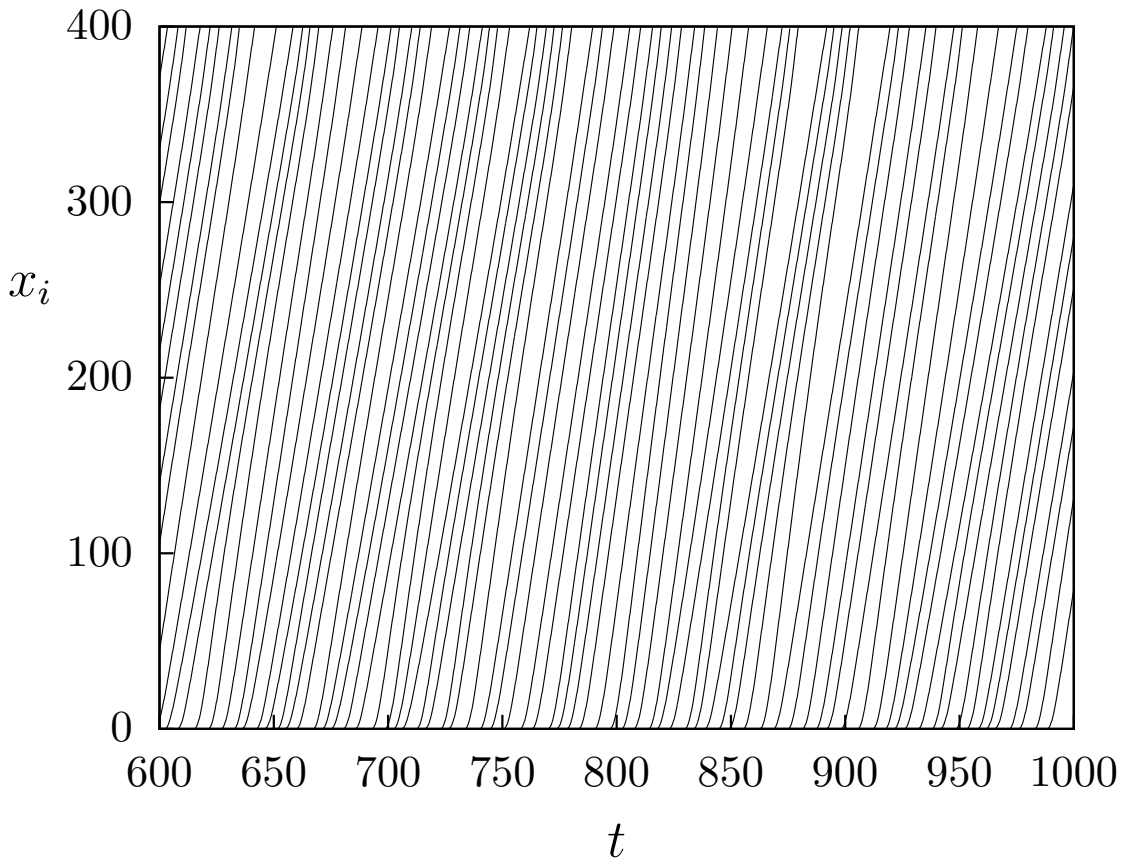
$$\frac{dv_i}{dt} = a_i \left[1 - \left(\frac{v_i}{v_i^*} \right)^{\delta} - \left(\frac{s_i^*}{s_i} \right)^2 \right] \ , \tag{14}$$

$$\frac{dx_i}{dt} = v_i \ . \tag{15}$$

```
    ...
  Glider.mess.addField("a", 2.8);
  Glider.mess.addField("b",0.9);
  Glider.mess.addField("g", 0);
  Glider.mess.addField("g0", 10.0);
  Glider.mess.addField("T",1.5);
//-------------------------------------------------------
  Glider.mess.addField("anterior",ant);
  if(ant != null )
    ant.addField("posterior", Glider.mess);
  Odes ode = new Odes(true, 0.0, 0.01);
  ode.addEqs(new vOde("vvx",0.0,Glider.mess));
  ode.addEqs(new xOde("x",Glider.mess.getdoubleValue("x")));
  Glider.mess.addField("ode",ode);
//-------------------------------------------------------
```

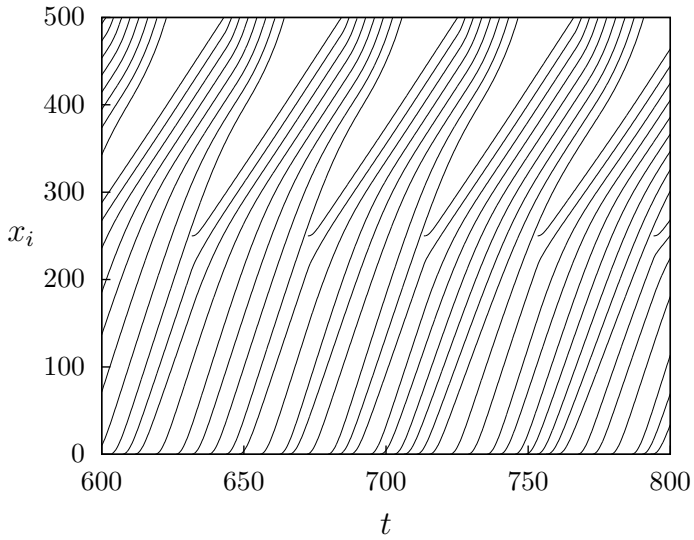
```
class vOde extends Cont {
      Message m;
    vOde(String l, double ic,Message m){
      super(l,ic);
      this.m = m;
    }
    public double feval(double x, Vector y){
            ...
      Message ant = (Message)m.getValue("anterior");
      if (ant != null) {
         x_ant = ant.getdoubleValue("x");
         v_ant = ant.getdoubleValue("vvx");
//       ------------------------------------------------------
         g = x_ant - x_m;
         gg = g0+v_m*T+v_m*((v_m-v_ant)/(2*Math.sqrt(a*b)));
         Vode = a*(1-Math.pow((v_m/v0),2)-(Math.pow((gg/g),2)));
      } else {
          Vode = a*(1-Math.pow((v_m/v0),2));
      }
      return Vode;
//    ------------------------------------------------------------
    }
}
class xOde extends Cont {
    Message m;
    xOde(String l, double ic, Message m){
        super(l,ic);
        this.m=m;
    }
    public double feval(double x, Vector y){
//      ------------------------------------
     double Xode = m.getdoubleValue("vvx");
        return Xode;
//      ------------------------------------
    }
```

Modelo base del condutor inteligente



Posición vs. tiempo

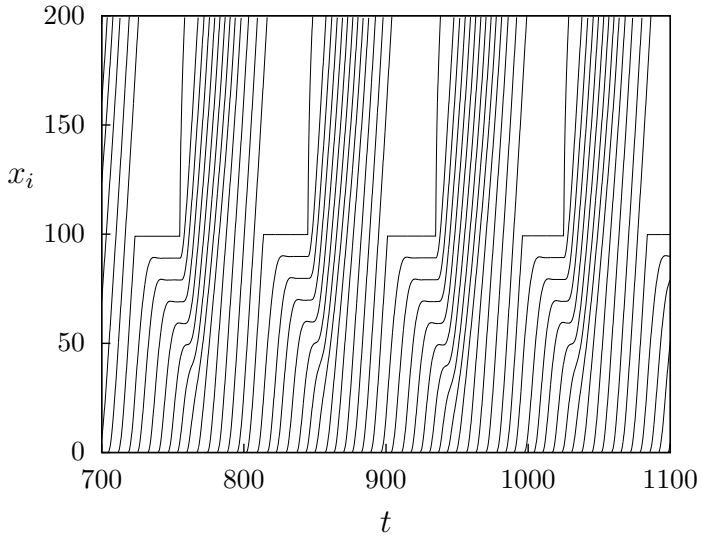
Incorporación



Posición vs. tiempo

```
import galatea.glider.*;
import galatea.gspaces.*;
class Incorporacion extends Node {
   static boolean activa;
   Message posterior;
   Incorporacion() {
//    ----------------------------
      super("incorporacion", 'A');
//    ----------------------------
      activa=false;
      posterior = null;
      Message m;
      int ll = Via.calle.getWait().getEl().ll();
      double x;
      for(int n=1; n<=ll;n++){
         m = (Message)Via.calle.getWait().getEl().getDat(n);
         x = m.getdoubleValue("x");
        if(x<=Via.l/2){
          Message a = (Message)(m.getValue("anterior"));
//        ------------------------------------------------
          if(x<Via.l/2-10&&(a==null||a.getdoubleValue("x")
    >Via.l/2+10)){
//        ------------------------------------------------
            posterior = m;
//          ----------------------------
            Glider.act(Via.entrada, 0);
            it(40);
//          ----------------------------
            activa=true;
            return;
          }else{
//          --------
            it(0.1);
            return;
//          --------
          }
        }
      }
      it(40);
      return;
   }
// -------------------------------
   public Message getPosterior(){
       return posterior;
// -------------------------------
```

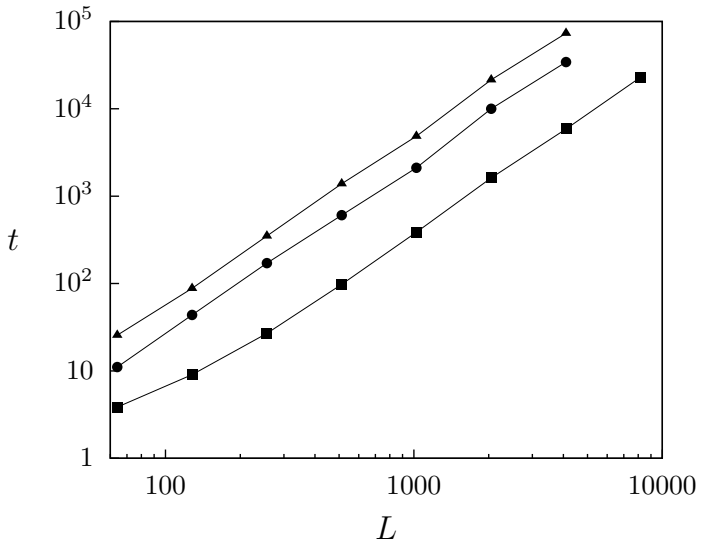
Semáforo




```
import galatea.glider.*;
```

```
class Semaforo extends Node {
    Semaforo(){
//      ----------------------
        super("semaforo",'A' );
//      ----------------------
        Glider.nodesl.add(this);
    }
    public void fact() {
//      ------------------------------------------
        Via.calle.getMove().getDoor(1)
                        .setClose(!Via.calle.getMove()
                        .getDoor(1).isClose());
//      ------------------------------------------
        if(Via.calle.getMove().getDoor(1).isClose()){
            it(35);
        }else{
            it(55);
    }
    }
}
```

Eficiencia de los modelos



tiempo(s) vs.longitud.

Leyenda

- ▲ Conductor inteligente
- ● Mapas acoplados
- ■ Autómatas celulares

$$t \sim L^{\alpha} \tag{16}$$

$$\alpha = 1{,}96 \pm 0{,}06 \tag{17}$$

Conclusiones y recomendaciones

- Se desarrolló e implementó los modelos de autómatas celulares, mapas acoplados y del conductor inteligente en GALATEA
- Se desarrollaron dos variantes; Incorporación y semáforo
- Se tomaron algunas medidas que resultaron como se esperaba.

Se recomienda;

- Hacer pruebas de solidez de los modelos, etc.
- Ajustar las cualidades físicas de los modelos como: velocidades, aceleraciones, longitud de la vía y otros, que producirían resultados más realistas.
- Aumentar el número de carriles.