Timed-pNets: A Formal Model for GALS systems

Yanwen Chen
Supervisors: Eric Madelaine Yixiang Chen

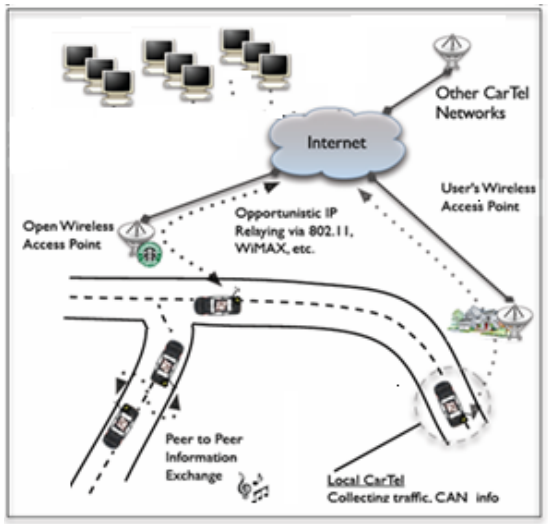
INRIA ECNU

28/03/2012

Outline

- Motivation
- Background
  - pNets
  - CCSL
- Our Solution
  - Timed-pNets : pNets + CCSL
  - syntax & semantic
- Current state/next steps

Motivation



(a)

(b)

- Various Embedded devices are connected by network(eg. Internet) for remote monitor/control,etc.
- For example: Intelligent Transportation Systems
  - Cars are equipped with sensors to know the car's speed, location, direction and also outside environment(traffic lights,etc).
  - Cars can communicate with each other.
  - Cars can communicate with Info center.

Scenarios of ITS

- **Local Control**
  - The speed of cars can be adapted by the distance between each other within a certain unit times
  - Cars must stop when traffic light turns red
- **Local communication**
  - Cars can communicate to each other
  - Communication among cars can be p2p or group communication
  - For example: Cars can ask for overtaking when emergent things happens.
- **Large scale communication**
  - Cars request info from info center and response are get within a certain time units.
  - Cars can report its location/states to info center.
  - Info center can communicate with cars to notice them the updated traffic info.
  - Info can forecast the future transportation and guide card to choose a best way to their destination.

System requirements and solutions

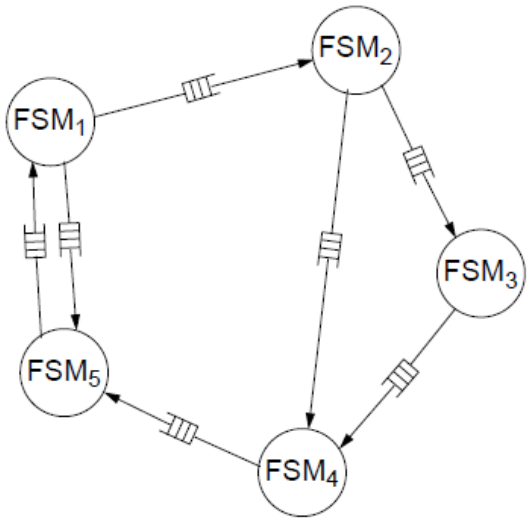
- System requirements:
  - **Boundary Safety:** If a behavior will happen before time expired
    e.g., $After!req_2^t(stop) Eventually?req_1^t(stop) \wedge (t_2 < 3)$
  - **Liveness:** If existing a path which can transit to another state in case time expired
- Possible solutions:
  - Introducing time constraints when building a formal model to describe its communication behavior
  - Reasoning about time properties of the formal model

asyn/syn model:

- An asynchronous model of a distributed system has no bounds on
  - process execution latencies
    – arbitrarily long but finite time may occur between execution steps
  - item message transmission latencies
    –message maybe received an arbitrarily long time after it was sent
  - clock drift rates
    –process's local clocks maybe arbitrarily unsynchronized
- A synchronous model, conversely, has a priori known upper and lower bounds on these quantities.

Ref: E.Douglas Jensen, Real-Time for the Real world

GALS model:



GALS: globally asynchronous locally synchronous

- Each FSM individually behaves like a synchronous systems
  - reacts instantaneously on a set of available inputs and generates output
- The global system is asynchronous
  - communication time is finite and non-zero
  - reaction time of each FSM, as viewed by other FSMs is finite and non-zero

Ref: Petru Eles, IDA, LiTH System Design&Methodologies

Parameterized Networks of Automata(pNets)

A semantic model for distributed systems. The advantages of pNets:

- parameterized and hierarchical behavioral model
- tree-like structure which leaves are LTS
- flexible enough for various programming structures, parallel constructions and communication modes.
- including data in model
- capable to build a finite model

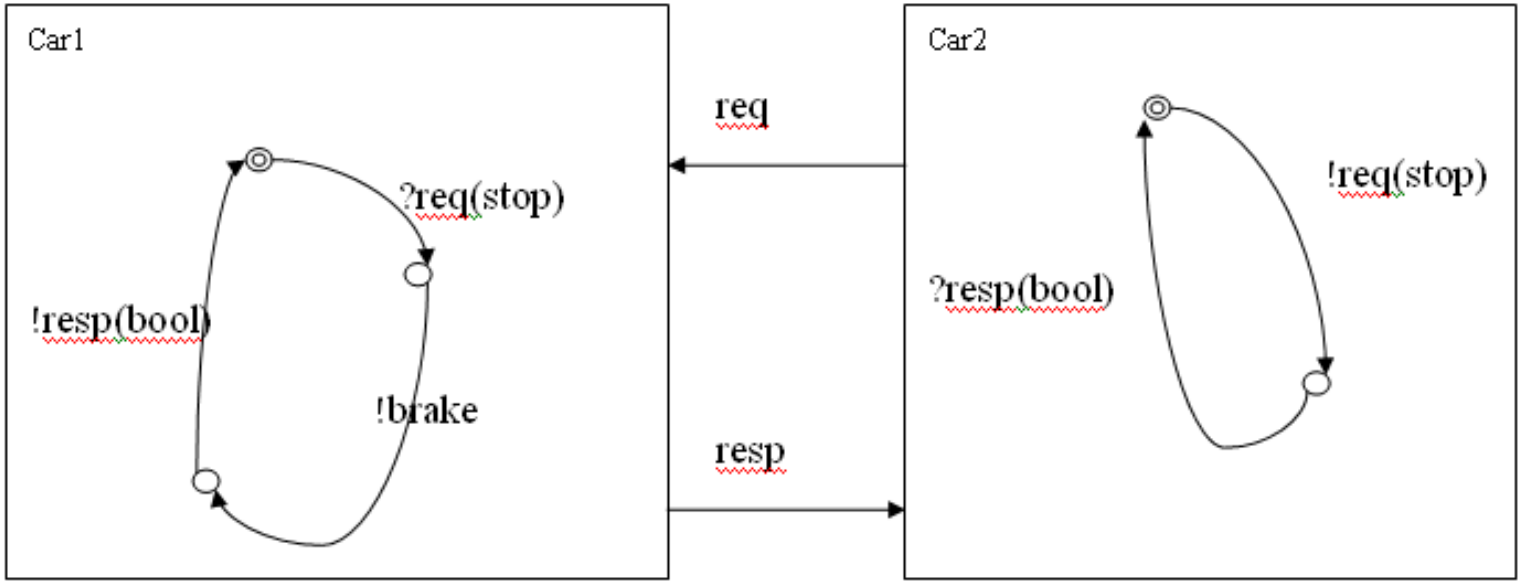
But time property is not including in it.
[Eric Madelaine, Oasis INRIA]

Clock Constraints Specification Language

- high-level time patterns to express multi-clock time specifications
- proposed a time model for real-time systems
- defined clock constraints of the time model

[Robert De Simone, Aoste INRIA]

Communication between cars



- each car is a FSM
- internal action & external action

Next step we will do is:

- introduce time delay into action
- introduce actions' clock constraints

Timed-Actions

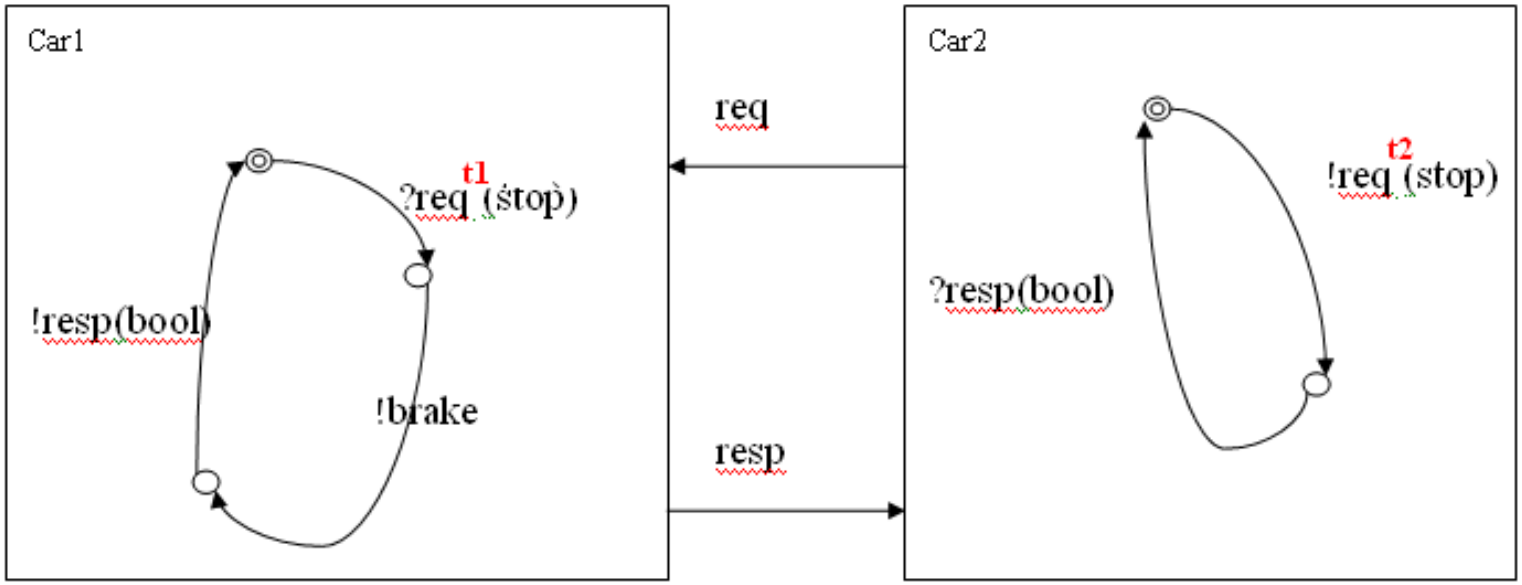
$\mathcal{T}$ is a set of discrete times taken from non-negative natural numbers $\mathcal{N}^+$, $\mathcal{L}_{\mathcal{A},\mathcal{T}}$ is a action set built over $\mathcal{T}$, which includes a constant action τ and observed actions each with a free time variable $t \in \mathcal{T}$. $a \in \mathcal{L}_{\mathcal{A},\mathcal{T}}$ is a timed action. The free time variable describes time delay before the action can be executed. $\mathcal{B}_{\mathcal{A},\mathcal{T}}$ the set of boolean expressions(guard) over $\mathcal{L}_{\mathcal{A},\mathcal{T}}$

For example: a^t means the action cannot be executed until t units times are passed.

Labeled transition system

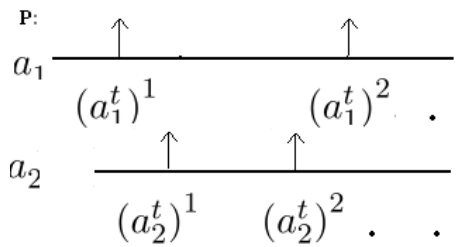
Definition: LTS is a tuple $< S, s_0, L, \rightarrow >$, where

- S(possibly infinite) is a set of states
- $s_0 \in S$ is the initial state
- L is set of labels
- $\rightarrow$ is the set of transition: $\rightarrow \subseteq S \mathrm{x} L \mathrm{x} S$. we write $s \xrightarrow{\alpha} s'$ for $(s, \alpha, s') \in \rightarrow$, in which $\alpha \in \mathcal{L}_{\mathcal{A},\tau}$.



clock definition

A clock C_a consists of a infinite set of discrete ticks of action a.
$C_a = \{(a^t)^1, (a^t)^2, ..., (a^t)^k, ...\}(k \in N, a^t \in \mathcal{L}_{\mathcal{A},\mathcal{T}})$.
$C_P = \{C_A\}$ presents the clock set of process P in which A is its action set, $a_i^t \in A \subseteq \mathcal{L}_{\mathcal{A},\mathcal{T}}$.
eg. If $\{a_1^t, a_2^t, ..., a_i^t, ...\} \in A$, then
$C_P = \{\{(a_1^t)^1, (a_1^t)^2, ...\}, \{(a_2^t)^1, (a_2^t)^2, ...\}...\}$



Definition: A Net is a tuple $< A_G, J, \widetilde{C}_J, \widetilde{O}_J, R, \overrightarrow{V} >$, where:

- $A_G \subseteq \mathcal{L}_{\mathcal{A},\mathcal{T}}$ is a set of global actions
- J is a countable set of argument indexes
- Each index $j \in J$ is called a hole and is associated with $O_j \subseteq \mathcal{L}_{\mathcal{A},\mathcal{T}}$
- C_j is a set of clocks in hole $j \in J$
- R is a set of clock constraints between actions in different holes.
- $\overrightarrow{V} = \{\overrightarrow{v}\}$ is a set of vectors of the form: $\overrightarrow{v} =< (a_g^{t_g}).\widetilde{(a^t)}_I >$, in which $a_g^{t_g} \in A_G, I \subseteq J \wedge \forall i \in I, {a_i}^{t_i} \in O_i, Sort(A_G, J, \widetilde{C}_J, \widetilde{O}_J, R, \overrightarrow{V}) \models R$

Clock Constraints

Definition: Clock constraints are clock relations that apply to two clock expressions.
The syntax of the relations include:

- $\subset$: subclock
- $\sharp$: exclusion
- $=$: coincidence
- $\prec$: strict precedence
- $\preceq$: precedence

Sort Constraints

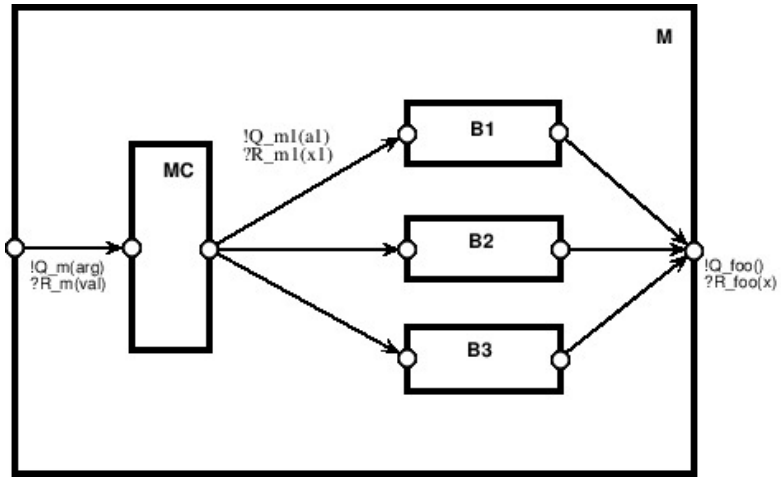
The Sort of a system is the set of actions that can be observed from outside of the system which conforms to the clock constraints. It is determined by its top-level node.

Definition: R is a set of clock relations,

$S = Sort(A_G, J, \widetilde{C}_J, \widetilde{O}_J, R, \overrightarrow{V})$, then $S \models R$ is defined by:

- $S \models \{c_{oi} \subseteq c_g | i \in (0,n)\} \Rightarrow \{a_g^{t_g} | a_g^{t_g} \in A_G, \{a_g^{t_g}, a_I^{t_I}\} \in \overrightarrow{v}, i \in I, t_g = max\{t_i\}\}$
- $S \models \{c_g \sharp c_{oi}\} \Rightarrow \{a_g^{t_g} | a_g^{t_g} \in A_G, \{a_g^{t_g}, a_J^{t_J}\} \in \overrightarrow{v}, j \in J, t_g = max\{t_j\}, j \neq i\}$
- $S \models \{c_g = c_{oi}\} \Rightarrow \{a_g^{t_g} | a_g \in A_G, t_g = t_i\}$
- $S \models \{c_{oi} \prec c_{oj} | i, j \in (0,n) \wedge i \neq j\} \Rightarrow \{a_g^{t_g} | t_i < t_j \wedge \{a_i, a_j\} \in \overrightarrow{v}, t_g = max\{t_i, t_j\}$

Example of Network of LTSs



$< A_G, J, \widetilde{C}_J, \widetilde{O}_J, R, \overrightarrow{V} >$ with:

- $A_G = \{!Q\_m^{tga}(arg), ?R\_m^{tgb}(val), !Q\_m1^{tgc}(a1), ...\}$
- $J = \{MC, B1, B2, B3\}$
- $R = \{c_{MC} \subseteq c_g, c_{B1} \subseteq c_g, c_{B2} \subseteq c_g, c_{B3} \subseteq c_g\}$
- $O_{MC} = \{!Q\_m^{tma}(arg), ?R\_m^{tmb}(val), !Q\_m1^{tmc}(al), ...\}$
- $O_{B1} = O_{B2} = O_{B3} =$
  $\{!Q\_m1^{ta}(al), ?R\_m1^{tb}(x1), !Q\_foo^{tc}(), ?R\_foo^{td}(), ...\}$
- $\overrightarrow{V} = \{$
  $<!Q\_m^{tga}(arg), !MC.Q\_m^{tma}(arg), -, -, - >$
  $<!Q\_m1^{tgc}(a1), !B1.Q\_m1^{tmc}(al), ?R\_m1^{tb}(x1), -, - >$

clock specification

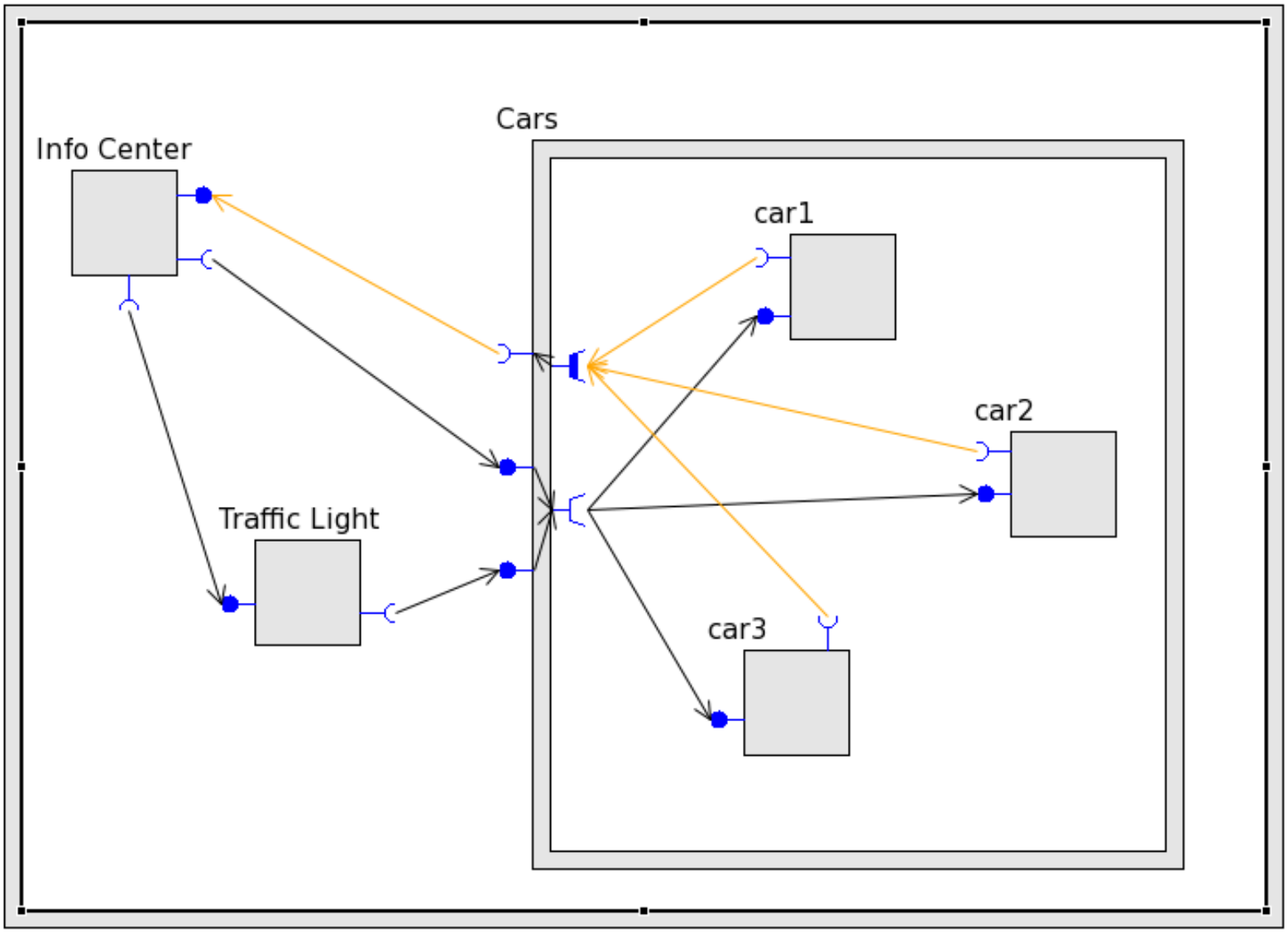
The clock specification is used to combining clocks and build abstract clocks.

- clock : clock reference
- $bool?clock : clock$: conditional expression
- $clock^{natural}$: await
- $clock + clock$: union
- $clock * clock$: intersection
- $clock \vee clock$: sup
- $clock \wedge clock$: inf

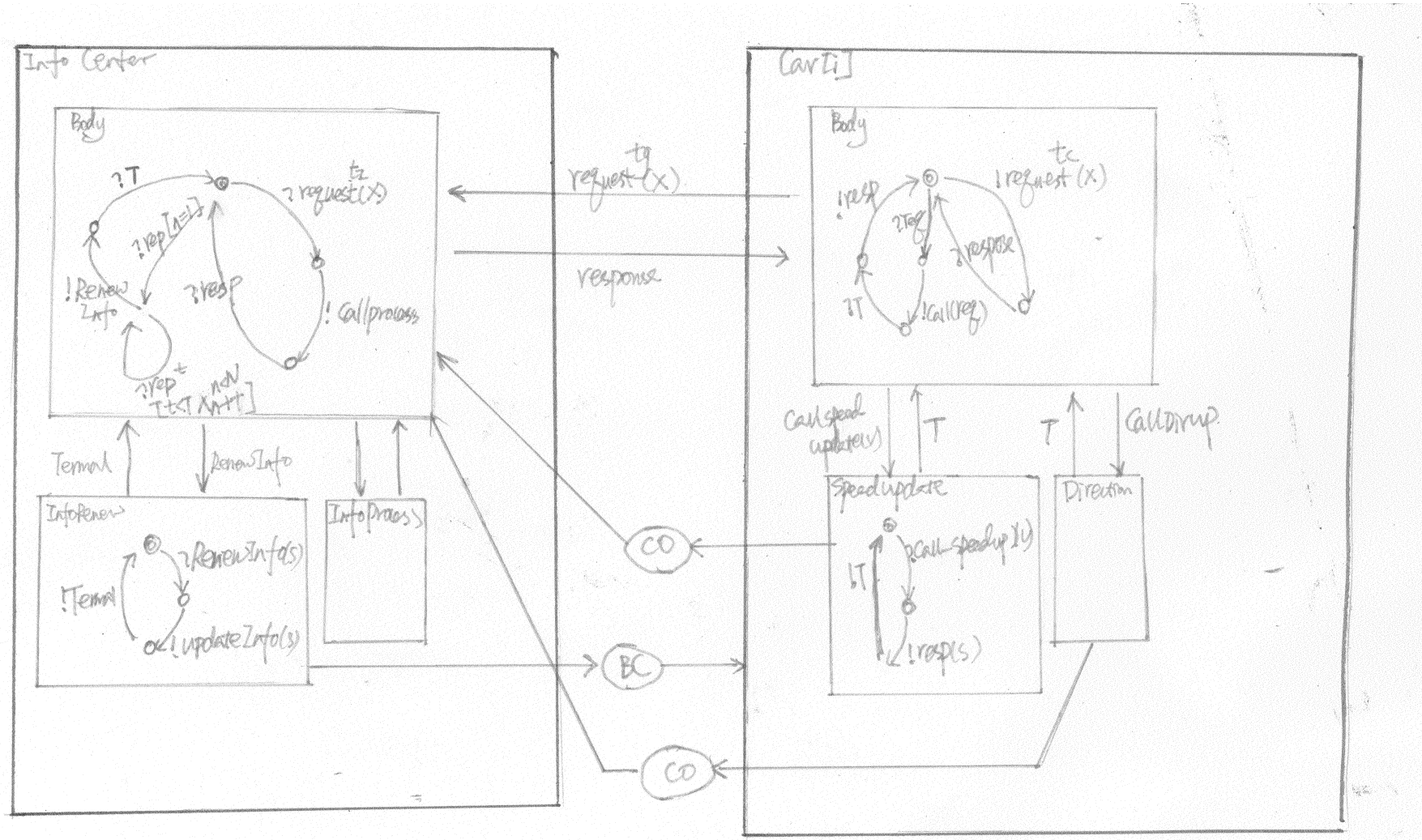
semantic of time definition constraints

- $[|c_o = c_M|] = \{\forall a \in c_o, \forall b \in c_M, a = b\}$
- $[|c_o = \beta ? c_M : c_N|] = \{c_o = ite(\beta, c_M, c_N)\}$
- $[|c_o = c_M^n|] = \{\forall a^t \in c_o, \forall b^t \in c_M, a^t = b^{t+n}\}$
- $[|c_o = c_M + c_N|] = \{c_o = c_M \vee c_o = c_N\}$
- $[|c_o = c_M * c_N|] = \{\forall a_o^t \in c_o, \forall a^{t_M} \in c_M, \forall a^{t_N} \in c_N, a_o = a, t = max\{t_M, t_N\}\}$
- $[|c_o = c_M \vee c_N|] = \{\forall a^t \in c_o, \forall a^{t_M} \in c_M, \forall a^{t_N} \in c_N, t_M < t_N ? a^t = a^{t_M} : a^t = a^{t_N}$
- $[|c_o = c_M \wedge c_N|] = \{\forall a^t \in c_o, \forall a^{t_M} \in c_M, \forall a^{t_N} \in c_N, t_M > t_N ? a^t = a^{t_M} : a^t = a^{t_N}$
- $[|c_o = c_M \prec c_N|] = \{\forall a^t \in c_o, \forall a^{t_M} \in c_M, \forall a^{t_N} \in c_N, a^{t_M} \prec a^{t_N}, t = (t_M + t_N)\}$

Model of ITS



Communication behavior of ITS



Current state & Next step

Current state:

- Introduce time delay into action
- Introduce time constraints into network of LTS
- Introduce time specification for building hierarchical network

Next Step:

- Develop 1 or 2 use-case scenarios
  - hypothesis clock
  - Proof of some clock properties
- implement the model
- checking the model

Reference

- E.Douglas Jensen, Real-Time for the Real world,
- Jose Meseguer and Peter Csaba Olveczky, Formalization and Correctness of the PALS Architectural Pattern for Distributed Real-time Systems
- Wang Yi, CCS + Time = an interleaving Model for real Time systems

Questions

Questions???

scenarios of local communication

- 4 cars with id: 1,2,3,4
- their speeds are: s1,s2,s3,s4
- sensor can check if the car are in the right position(successful to changing its direction:(1: yes, 0:no)).
- sensor can know the other cars speed, location, direction if the distance bt them in a certain circle and register these cars.
- sensors can check its own speed, location and direction.
- Assume they are moving on the same direction.
- car 1 want change its way from road 1 to road 2.
- car 1 send request to car 2,3,4 to notify them to adapt their speed
- car 2,3,4 response its requests and send back its speed info to car 1.
- when all three cars are agree to adapt their speed and send back the resp to car 1, then car 1 start to change way.
- request: request will be send each 2 sec, response will be