# Design Decisions

## Basics

* Each Actor can only process on message at a time.
* Because computation delay is negligible compare to network delay, statements in a message server is executed coarsely until an explicit delay statement is reached.
* At most one Continuous behavior can be active on each continuous variable. ~~This restriction is due to tooling (SpaceEx) and compositional semantic.~~ This is due to our hybrid automata-like syntax
* Continuous actions (timed actions included) can only be executed when no instantaneous actions is possible.
* Each actors defines a set of known rebecs that can communicate with. The communication media (CAN or wire) is define on the instantiation of the actor by annotating the actual known rebecs.
* Self-messages are always sent directly (not with CAN).
* Network delays and priorities for each messages must be defined.
* Internal message scheduling in actors?!!!

## Separation of Cyber actors from physical actors

Physical actors are the actors that process physical behaviors. They contain a set of continuous variables and a set of modes. The modes define the continuous behavior on the variables. A physical actor can only be one mode at each time. Physical actors can execute actions on the mode expiration. These actions include sending a message to cyber actors and resetting a continuous variable. Physical actors can read-share their variables. These actors can’t have delayed statements.

Cyber actors are the actors that process cyber behaviors. They contain a set of discrete variables and a set of methods (message servers). Cyber actors can communicate which other by sending (asynchronous) messages. Cyber actors can change the physical actors’ mode (synchronous).

## CAN network

CAN is a serial priority based network, where the priority of messages are unique. We assumed the network execution is not instantaneous, so CAN is executed when there is no possible instantaneous actions. When a messages need to be sent through CAN, the message will buffered in the network when all possible instantaneous actions are done. Then the network will choose the highest priority messages and until the messages is delivered (meaning it waits for the specified network delay of than messages and then it puts the messages in the destinations actor). The algorithm is executed in the next round of network.

# Features of HPalang (Hybrid Rebeca) are

* Formal semantics based on Hybrid Automata
* Asynchronous message passing between actors
* Computation time modeling
* Physical/Continuous behavior modeling
* Discrete behavior modeling

# Semantics

DVar : set of all discrete variables names.

CVar : set of all continuous variables names.

Mtd : set of all method declarations.

Mode: set of all modes.

Each method is defined as the tuple .

Each mode is defies as



and are invariant of behavior, flow of continuous variables and the guard of transition (expiration of the mode) and are a sequence of send or cassignment statement which will be executed when transition is taken.

# Statements





# Auxiliary functions

in which D(x) returns the delay variable for actor ID.

in which bod returns the body of method in actor .

# Operational Semantics

The global state is a tuple where is a function and

CS is a function

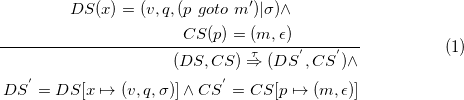
# Transitions

***Message Take***

***//Retain the labels of***

**Network Director**

***GoTo***



***Continuous Behavior Expiration***

***//Execute the actions here***

***Continuous Variable Assignment***

***Discrete Variable Assignment***

***Conditional True***

***Conditional False***

***Resume Statement***

***Continuous Behavior Statement***

***Delay Statement***

***Message Send***

**If message will be invalidated exactly on deadline**

**Hybrid Translation**

Preprocess on LTS

1. Prioritizing transition: if a state has at least one transition all its guarded transitions are removed
2. Merging transition: Merging transition so only guarded transition are left

From ) and CVar To

**Simplifications and assumptions**

No Parameter for methods.

Only one guard and invariant in continuous behavior.