# Design Decisions

* Each Actor can only process on message at a time.
* Simultaneous continuous behavior in one actor is allowed.
  + Equations on same variable are summed for continuous behavior.
* There is only on type of message and this message can contain continuous and computational behavior at the same time.
* Because most continuous behaviors have physical nature
  + Processing a continuous transition (behavior deprecation) has higher priority than processing normal message
  + Continuous transitions can be processed even when actor is suspended (due to computation delay)
* Continuous transitions cannot have time delays.
* Because computation delay is negligible compare to network delay, statements in a message server is executed atomically until an explicit delay statement is reached.

# Semantics

DVar : set of all discrete variables names.

CVar : set of all continuous variables names.

Mtd : set of all method declarations.

Cap: Capacity of message queue

Each method is defined as the tuple .

# Statements

# Auxiliary functions

inv: in which inv(cb) returns the invariant of continuous behavior cb.

guard: in which guard(cb) returns the guard of continuous behavior cb.

ode: in which ode(cb) returns the ordinary differential equations of continuous behavior cb.

action:in which action(cb) returns the transition statements of continuous behavior cb.

D : in which D returns the delay variable for actor ID.

QueueCap : : in which Cap returns the queue capacity for actor ID.

Body:

# Operational Semantics

The global state is a function and DS is the discrete state and is defined as } and HS is the continuous state and is defined as

# Transitions

For simplicity discrete variables are omitted.

## Low Priority Message Take

## High Priority Message Take

## Resume Statement

## Continuous Behavior Expiration

## Continuous Variable Assignment

## Continuous Behavior Statement

## Delay Statement

## Message Send

## After Message Send

## Scheduler Send statement

**Hybrid Translation**

From ) and CVar To

**Simplifications and assumptions**

No Parameter for methods.

Only one guard and invariant in continuous behavior.

Issues:

* What happens when two actor are ready to process their message?
* What happens when there are more than one message to be sent from an actor? Are they sent atomically? What about when there is a delay statement between them?