Semantic-Based Modelling of Embedded Adaptive Systems

Ina Schaefer
Software Technology Group
TU Kaiserslautern
inschaef@informatik.uni-kl.de

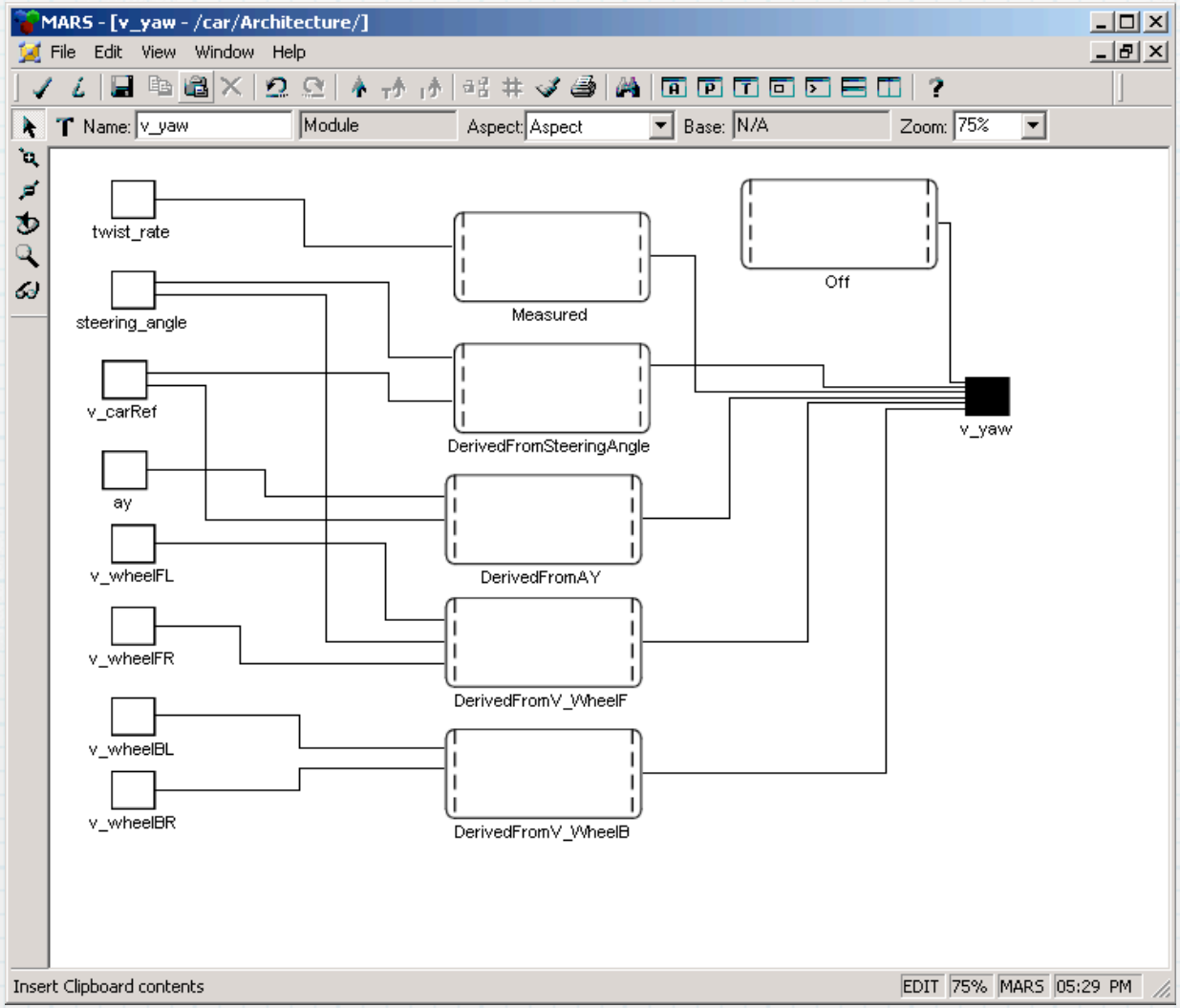
(joint work with Arnd Poetzsch-Heffter)

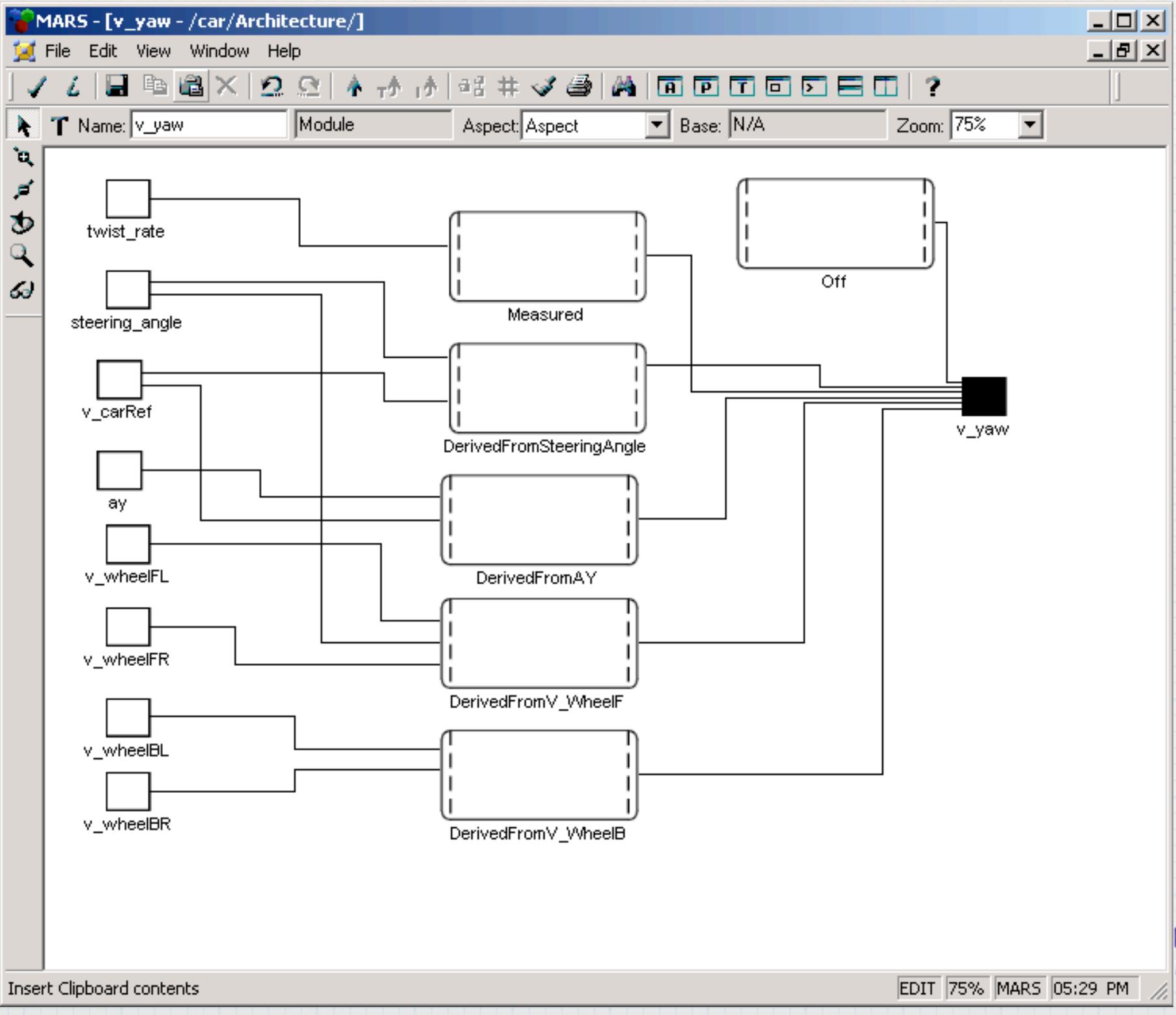
DASMOD

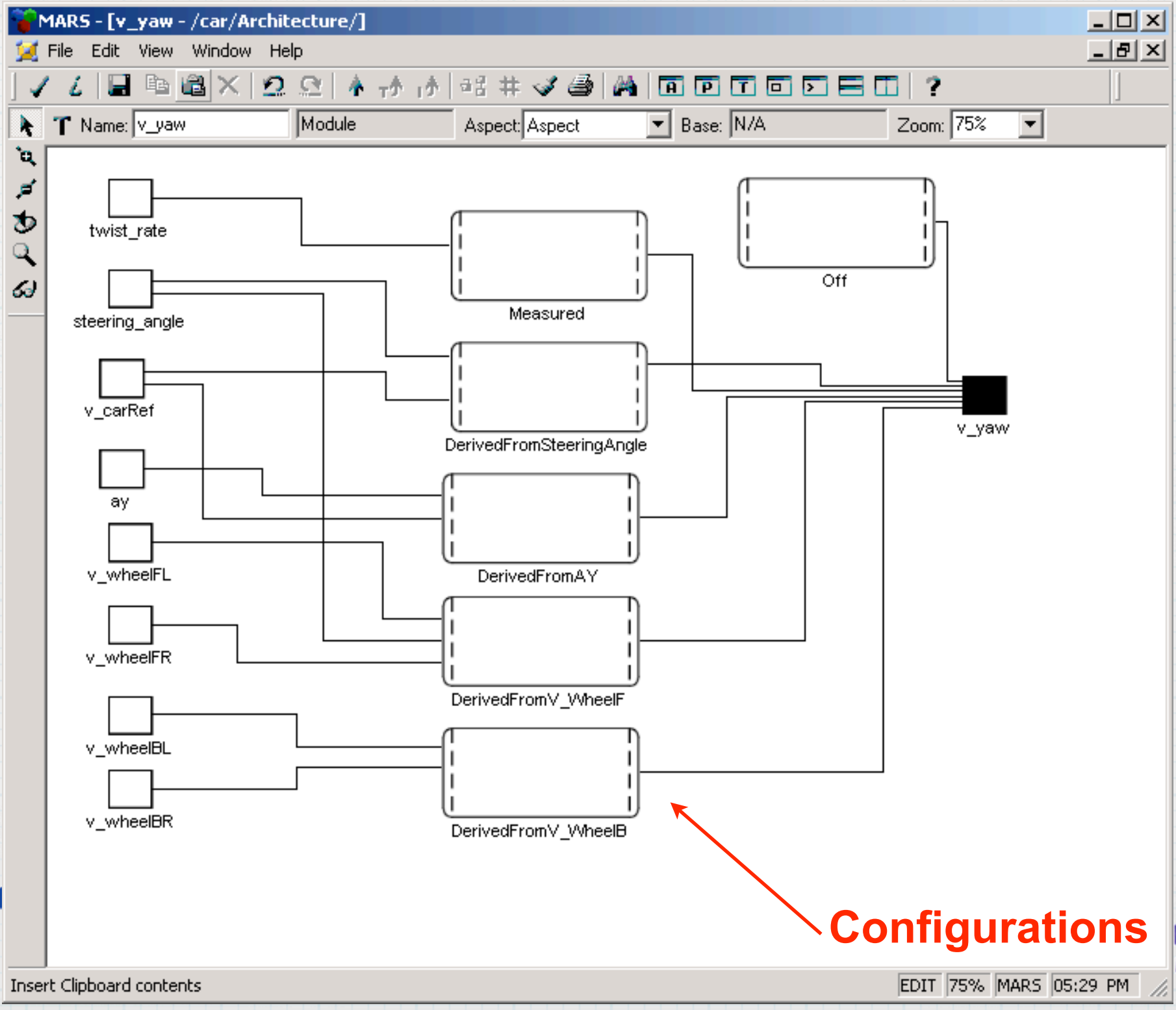
Embedded Adaptive Systems

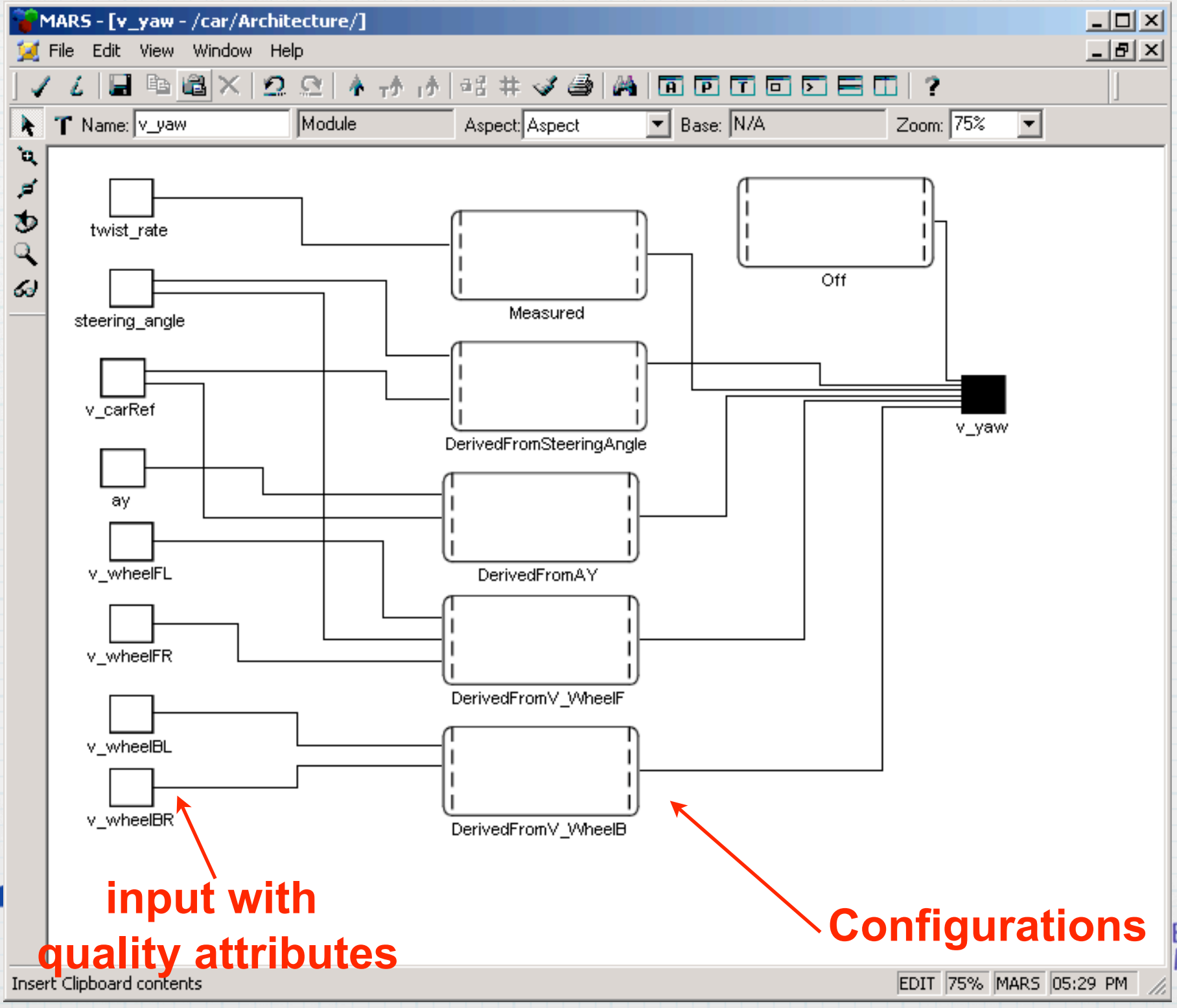
- Adaptation of behaviour according to environment
- Hence, system development is highly complex
- Support for system modelling and verification required

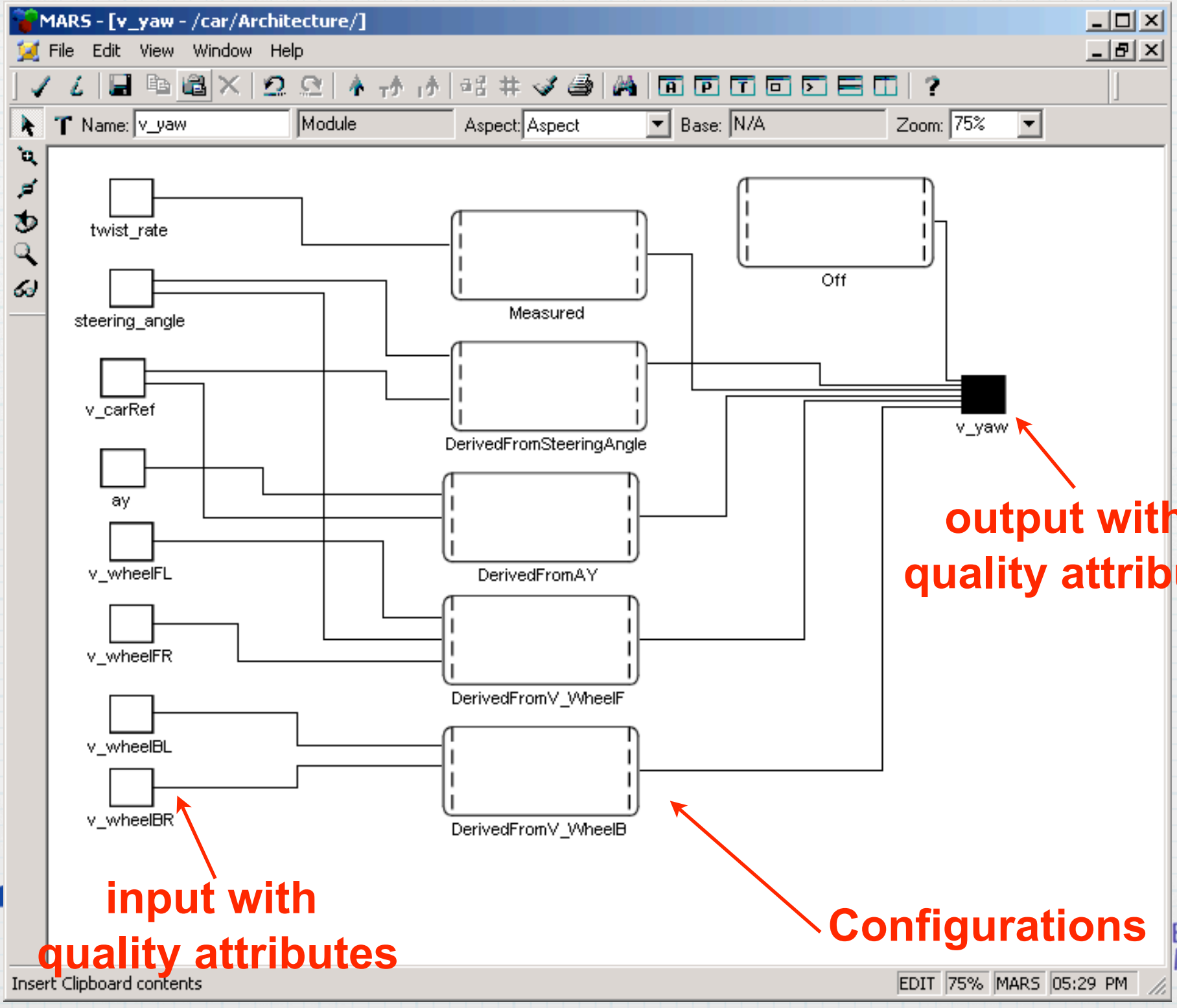
GME Model of CoCar




MARS - [v_yaw - /car/Architecture/]
File Edit View Window Help
Name: v_yaw
Module
Aspect: Aspect
Base: N/A
Zoom: 75%
twist_rate
steering_angle
v_carRef
ay
v_wheelFL
v_wheelFR
v_wheelBL
v_wheelBR
Measured
Off
DerivedFromSteeringAngle
DerivedFromAY
DerivedFromV_WheelF
DerivedFromV_WheelB
v_yaw
Insert Clipboard contents
EDIT
75%
MARS
05:29 PM


MARS - [v_yaw - /car/Architecture/]
File Edit View Window Help
Name: v_yaw
Module
Aspect: Aspect
Base: N/A
Zoom: 75%
twist_rate
steering_angle
v_carRef
ay
v_wheelFL
v_wheelFR
v_wheelBL
v_wheelBR
Measured
Off
DerivedFromSteeringAngle
DerivedFromAY
DerivedFromV_WheelF
DerivedFromV_WheelB
v_yaw
Configurations
Insert Clipboard contents
EDIT
75%
MARS
05:29 PM


MARS - [v_yaw - /car/Architecture/]
File Edit View Window Help
Name: v_yaw
Module
Aspect: Aspect
Base: N/A
Zoom: 75%
twist_rate
steering_angle
v_carRef
ay
v_wheelFL
v_wheelFR
v_wheelBL
v_wheelBR
Measured
Off
DerivedFromSteeringAngle
DerivedFromAY
DerivedFromV_WheelF
DerivedFromV_WheelB
v_yaw
input with quality attributes
Configurations
Insert Clipboard contents
EDIT 75% MARS 05:29 PM


MARS - [v_yaw - /car/Architecture/]
File Edit View Window Help
Name: v_yaw
Module
Aspect: Aspect
Base: N/A
Zoom: 75%
twist_rate
steering_angle
v_carRef
ay
v_wheelFL
v_wheelFR
v_wheelBL
v_wheelBR
Measured
Off
DerivedFromSteeringAngle
DerivedFromAY
DerivedFromV_WheelF
DerivedFromV_WheelB
v_yaw
output with quality attribute
input with quality attributes
Configurations
Insert Clipboard contents
EDIT 75% MARS 05:29 PM

Verification Techniques

Simulation of Adaptation

GME/ MARS

Theorem Prover

Quartz

Manual Verification of Adaptation Behaviour
[Schneider,Schuele,Trapp:2006]

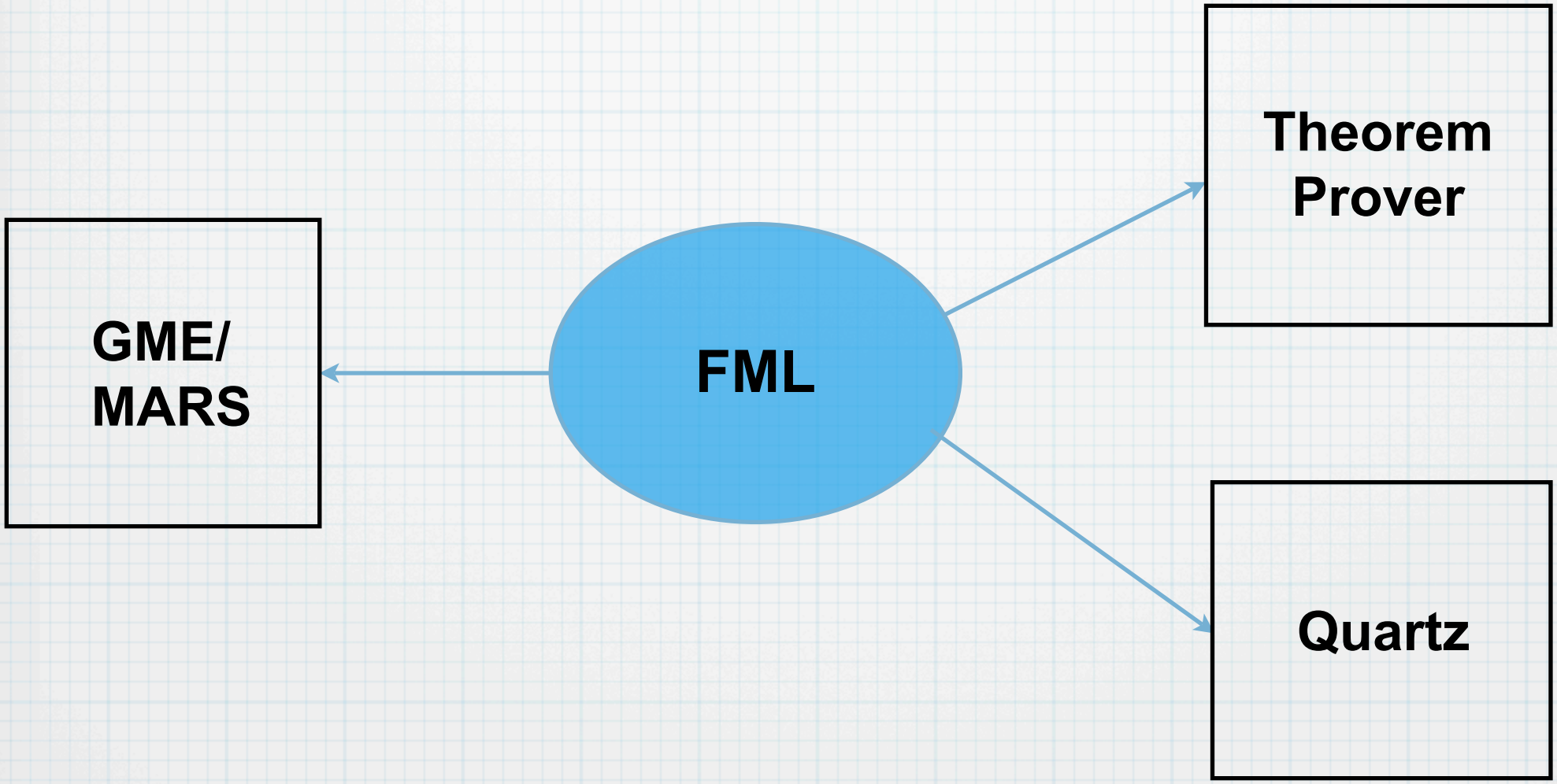
Our Workplan

Theorem Prover

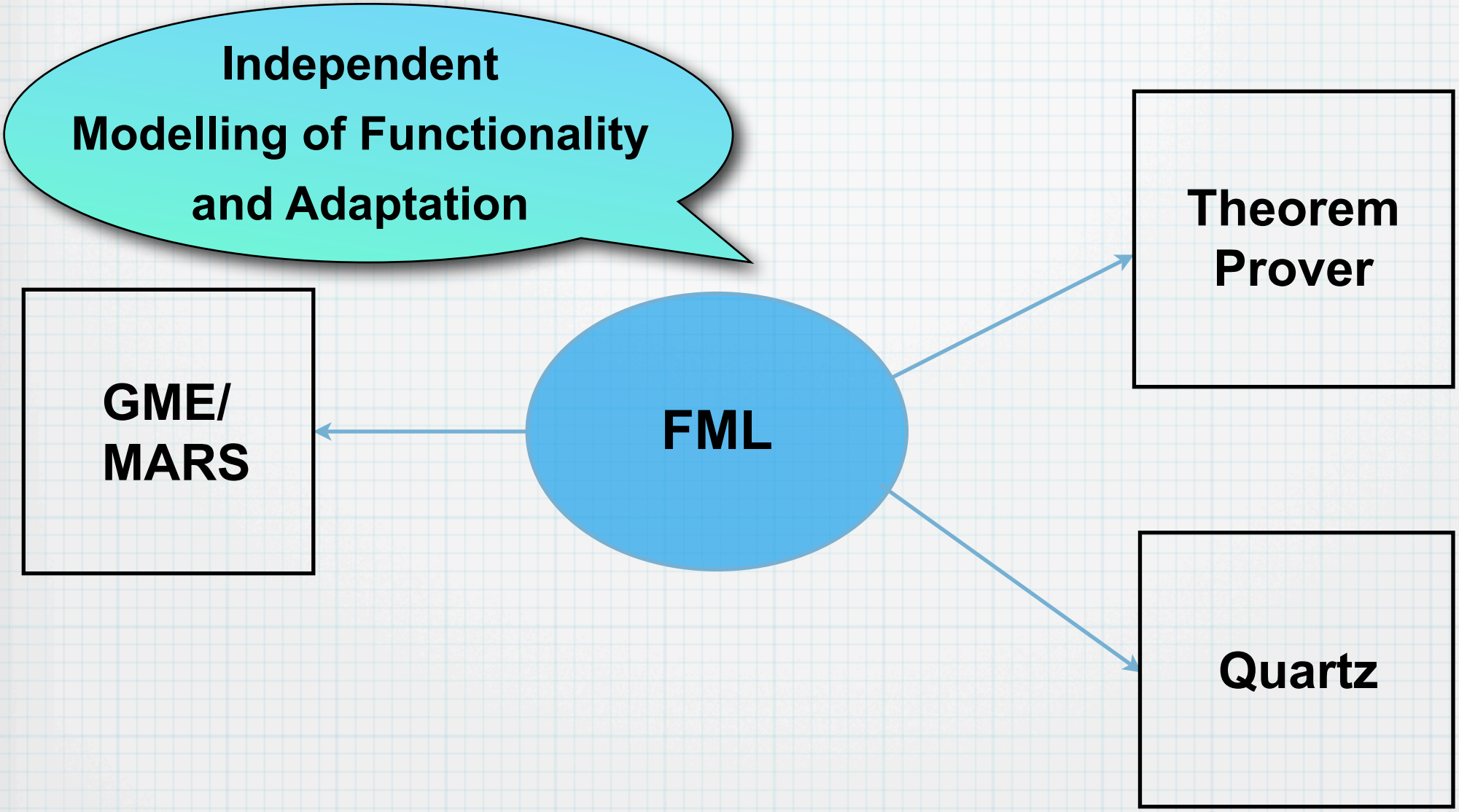
GME/ MARS

Quartz

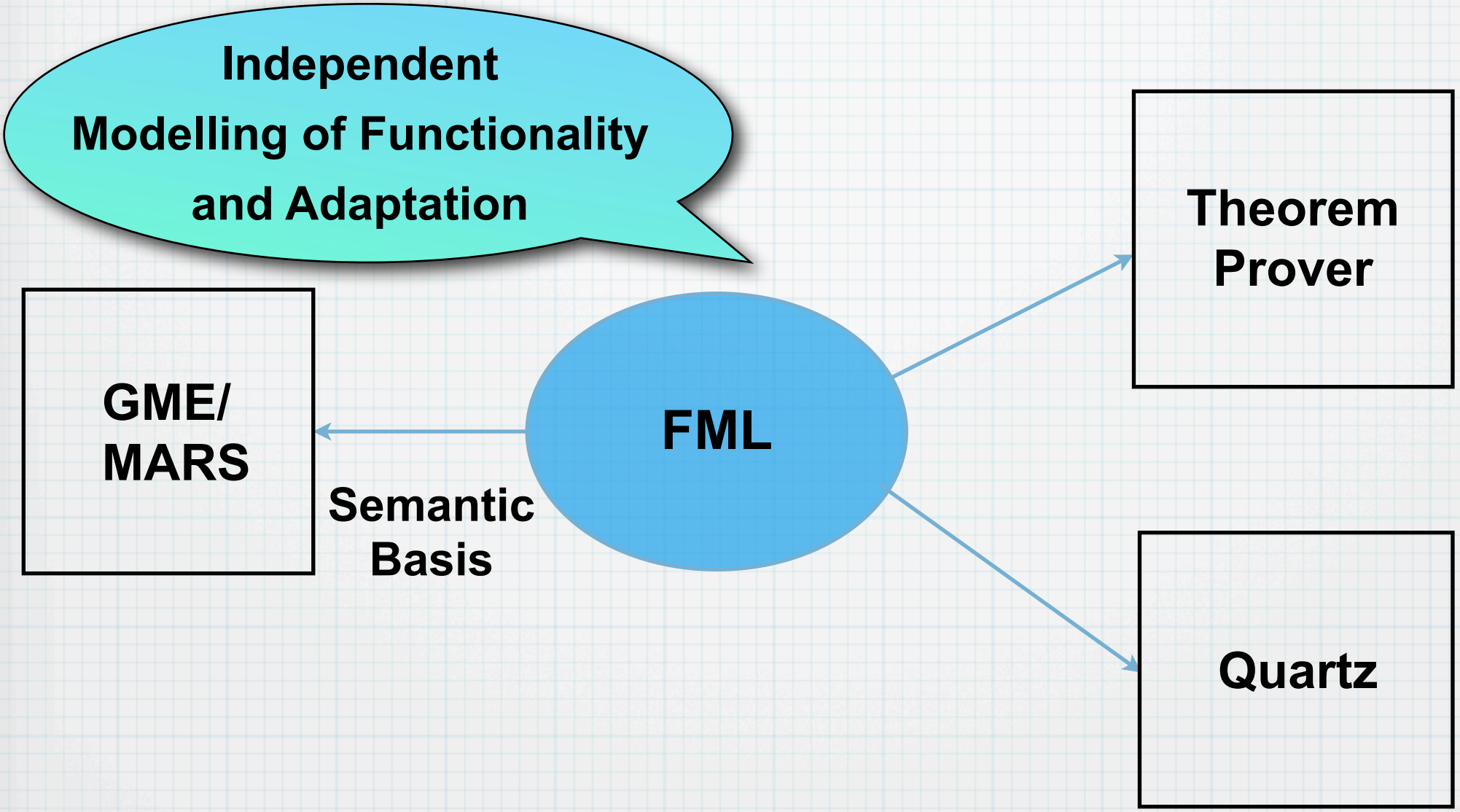
Our Workplan



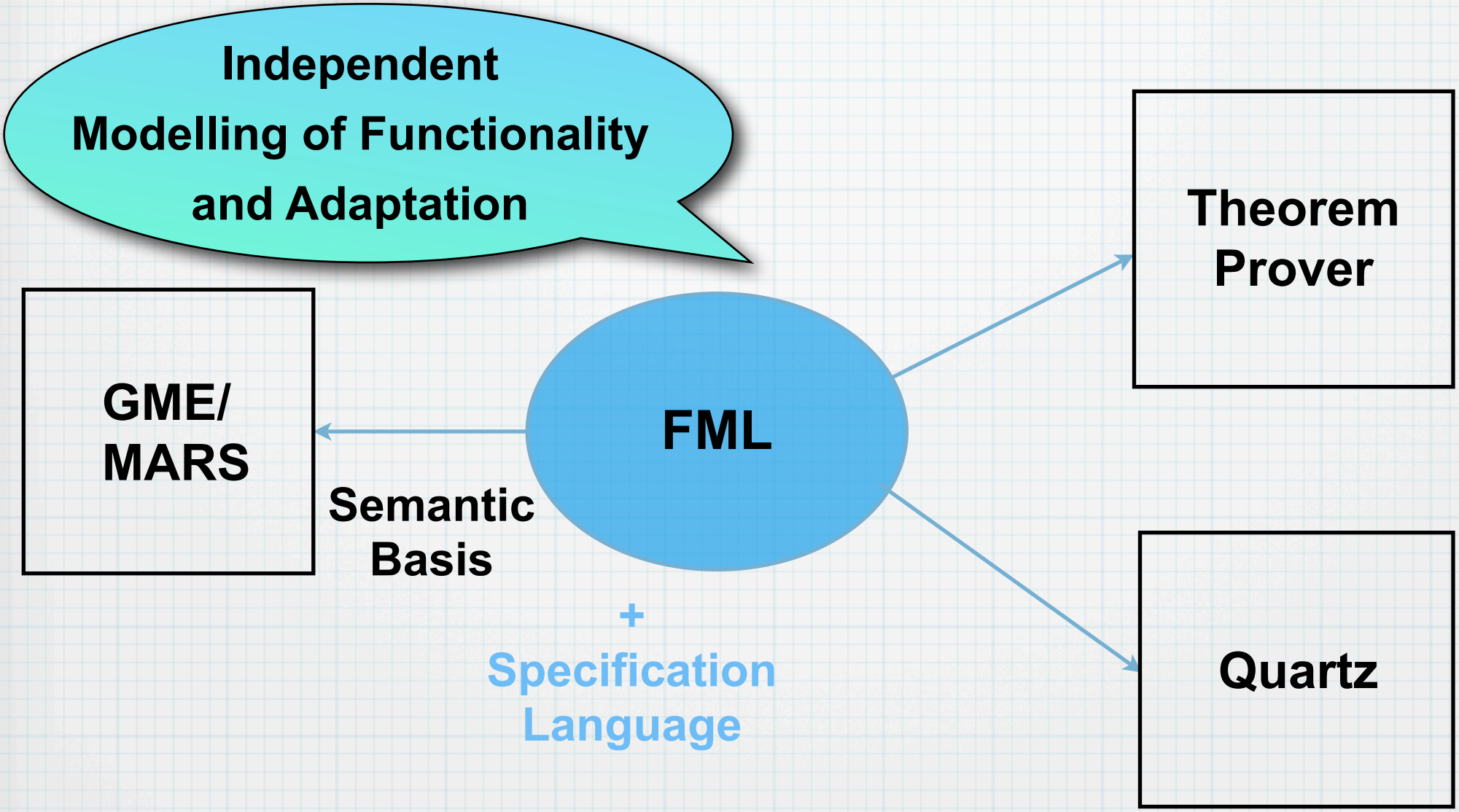
Our Workplan



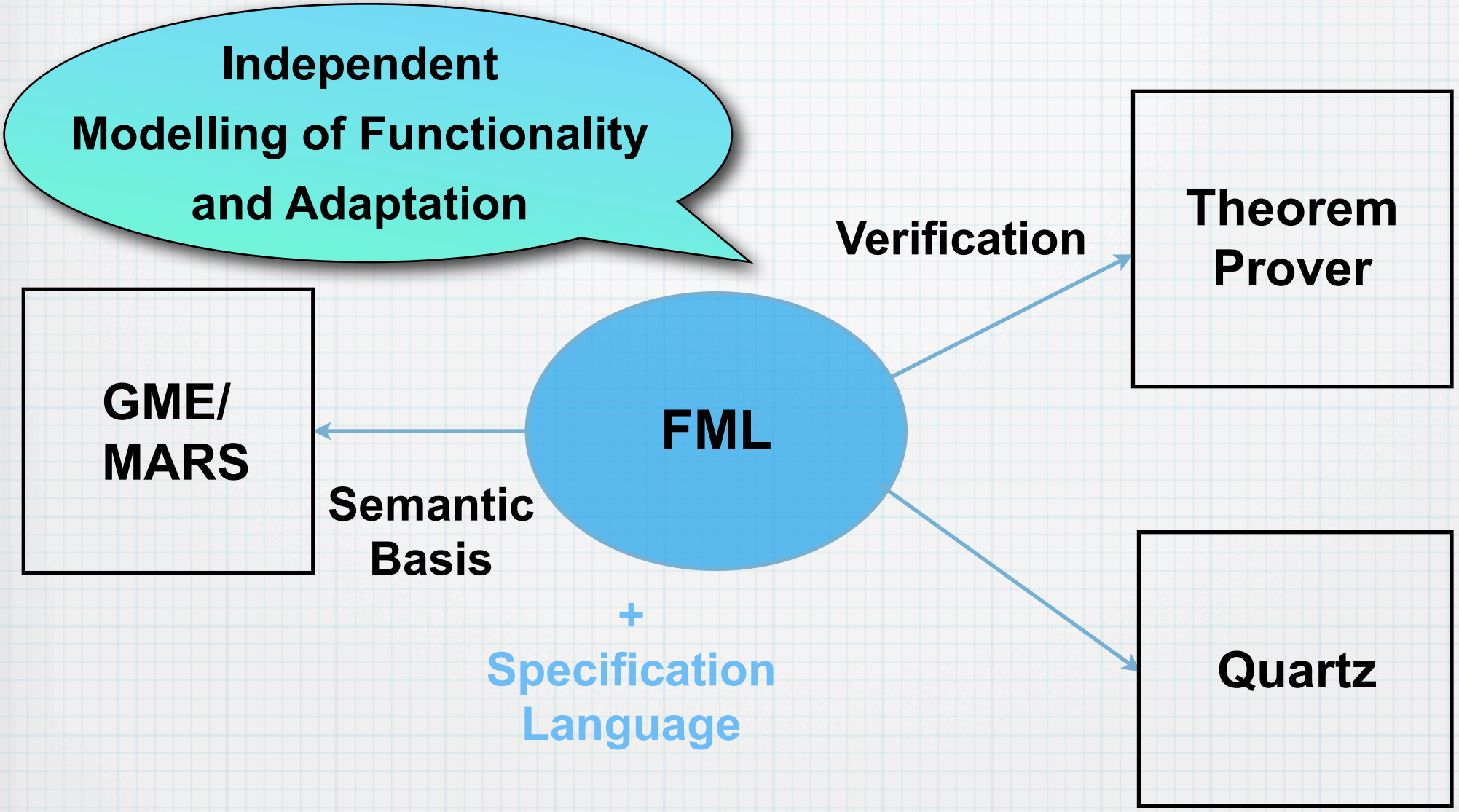
Our Workplan



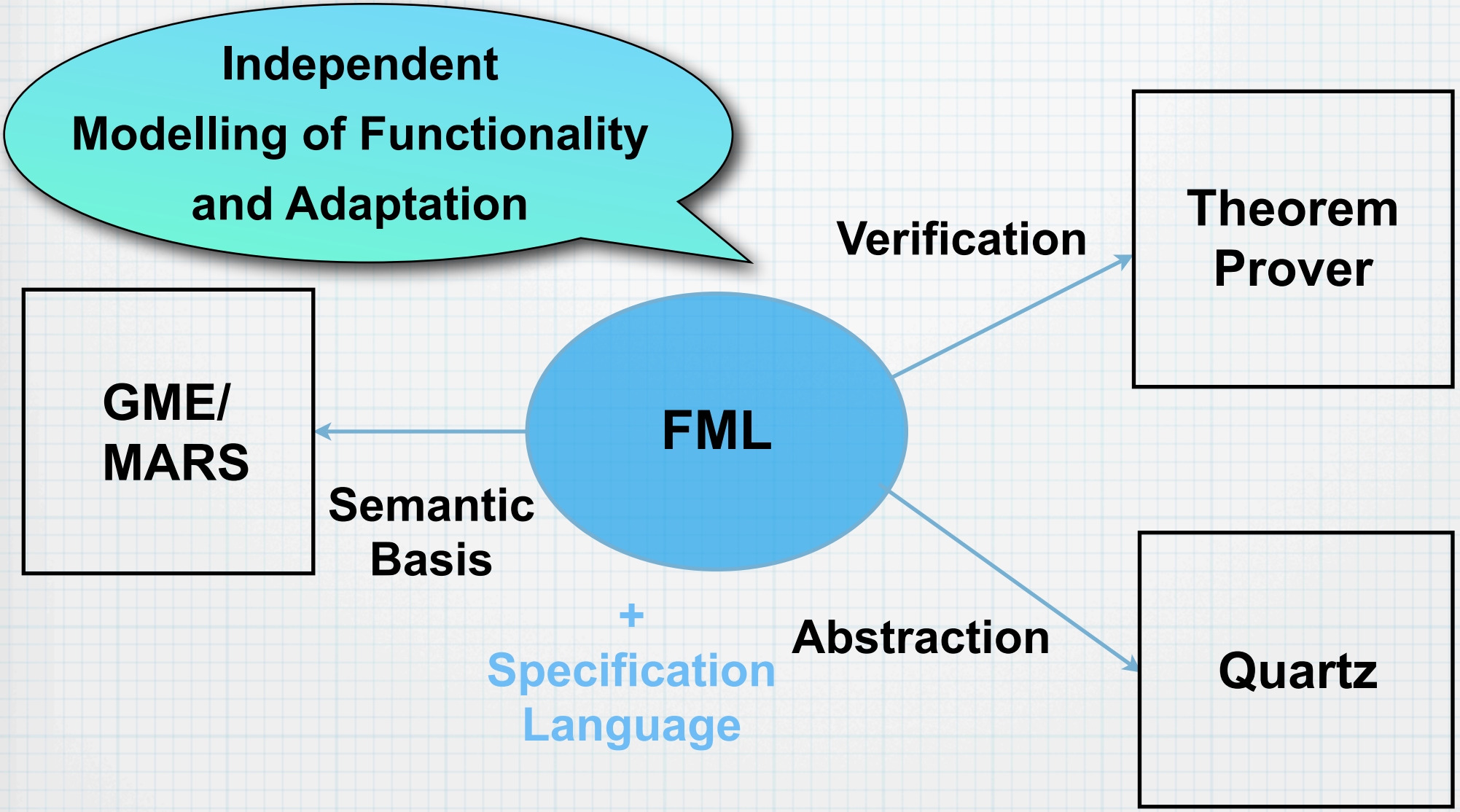
Our Workplan



Our Workplan



Our Workplan



Outline

- FML Formal Modelling Language
  - Syntax
  - Semantics
  - Case Study
- Specification Language
- Abstraction
- Summary and Future Work

FML

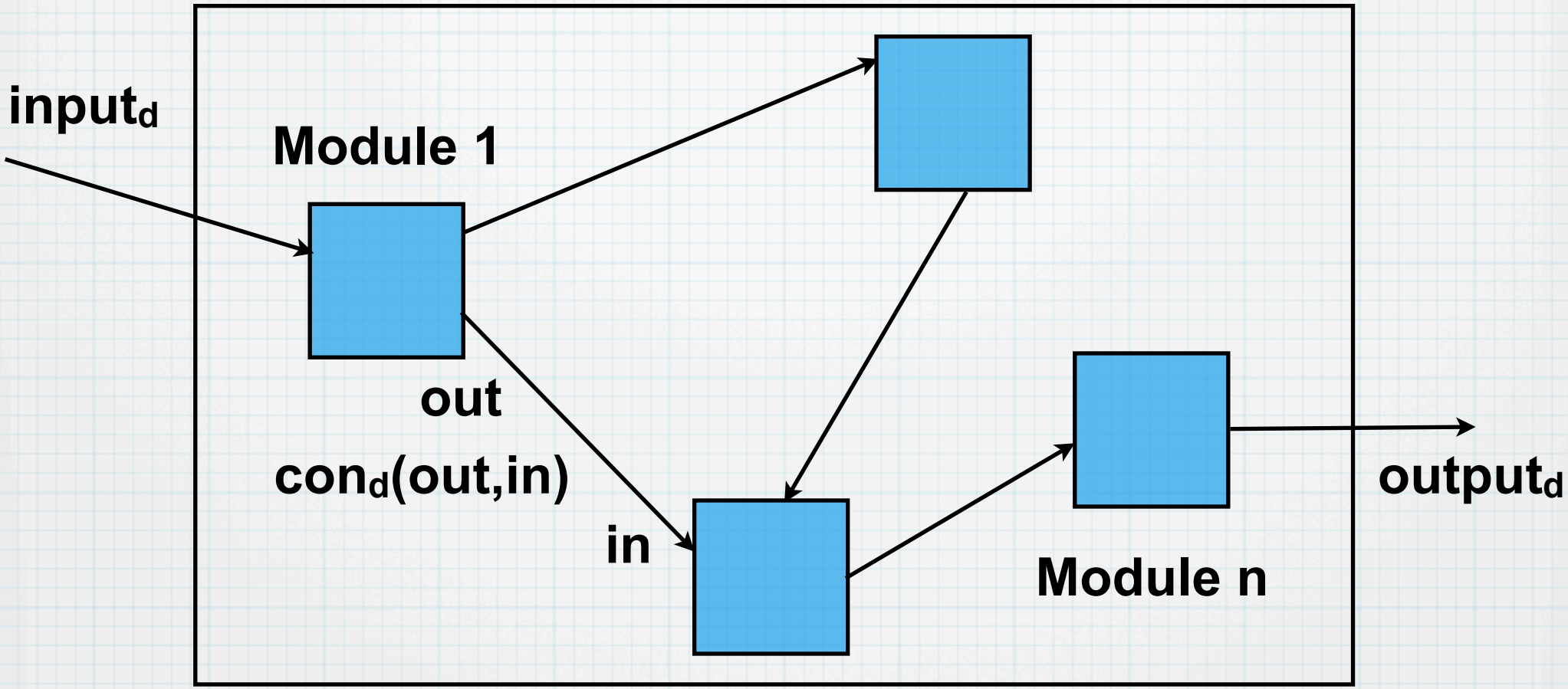
Design Goals:

- Explicit and Separated Modelling of Adaptation and Functionality
- Modular and Hierachical Specification

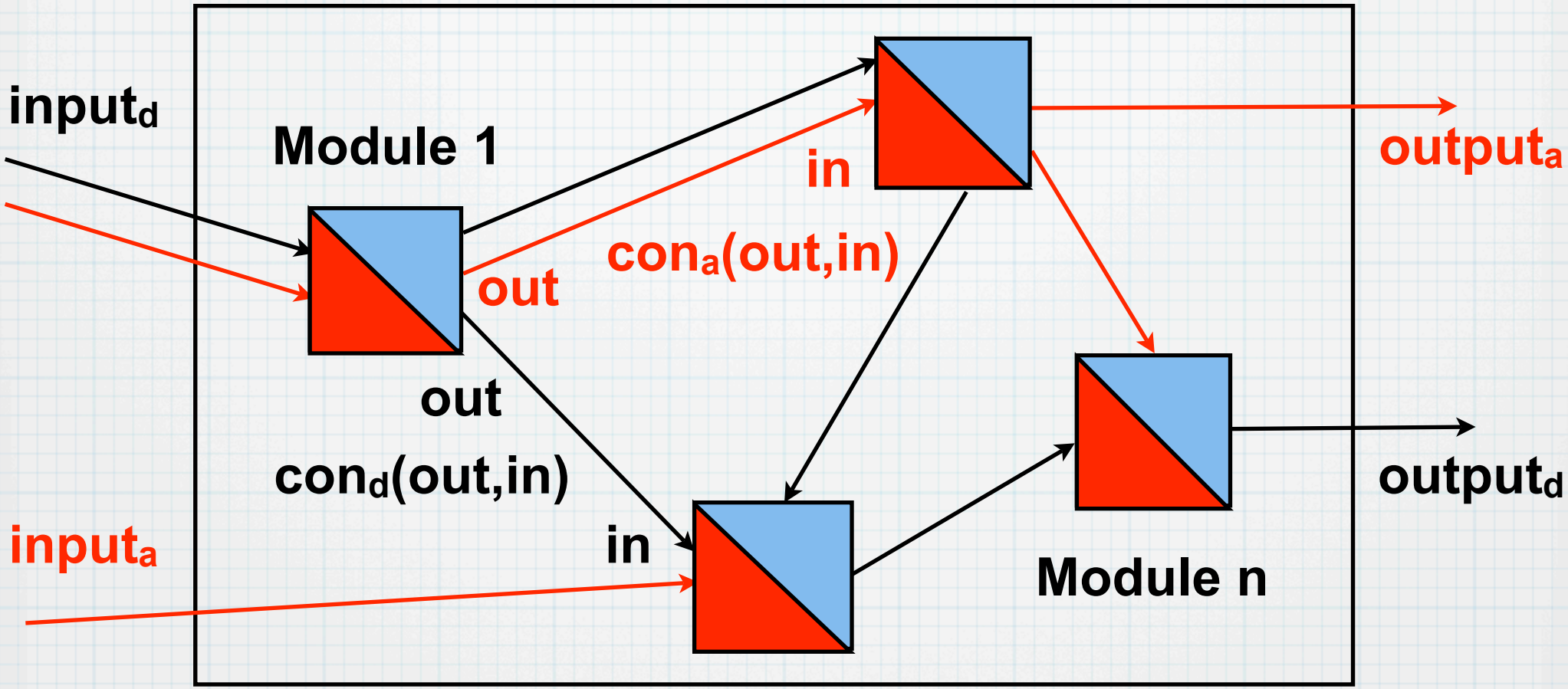
Main Concepts:

- Based on State-Transition Systems
- Uses Adaptation Aspect

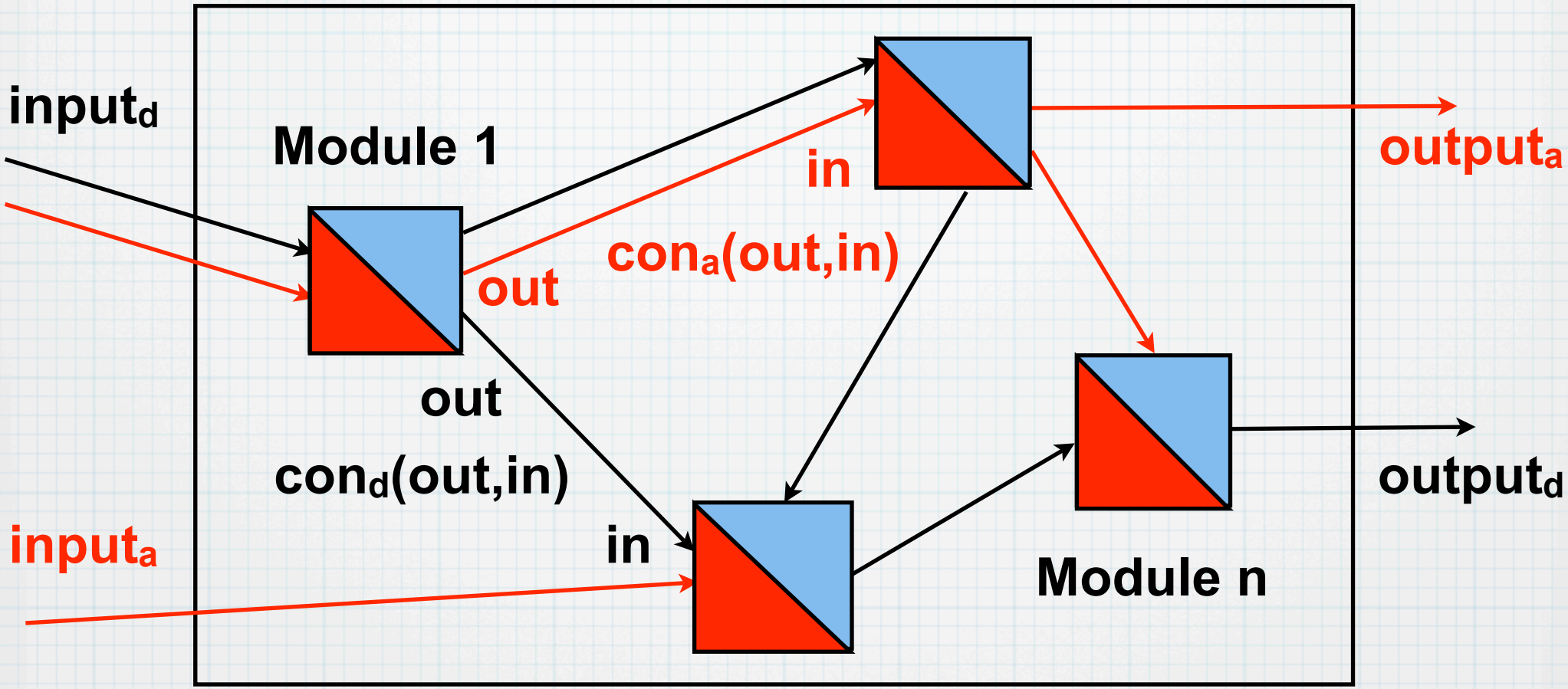
FML - System



FML - System


input$_d$
Module 1
in
output$_a$
con$_a$(out,in)
out
out
con$_d$(out,in)
input$_a$
in
Module n
output$_d$

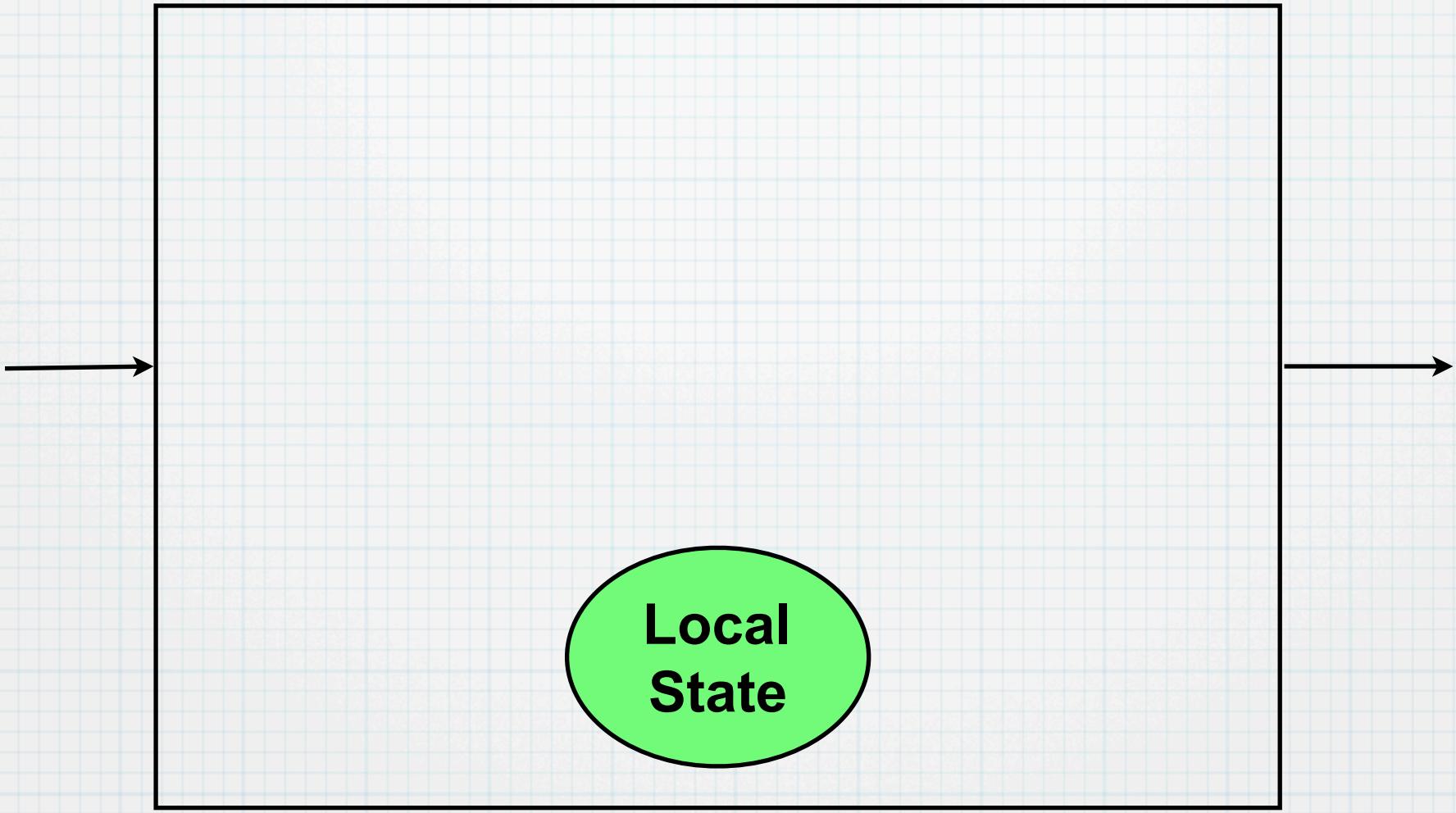
FML - System



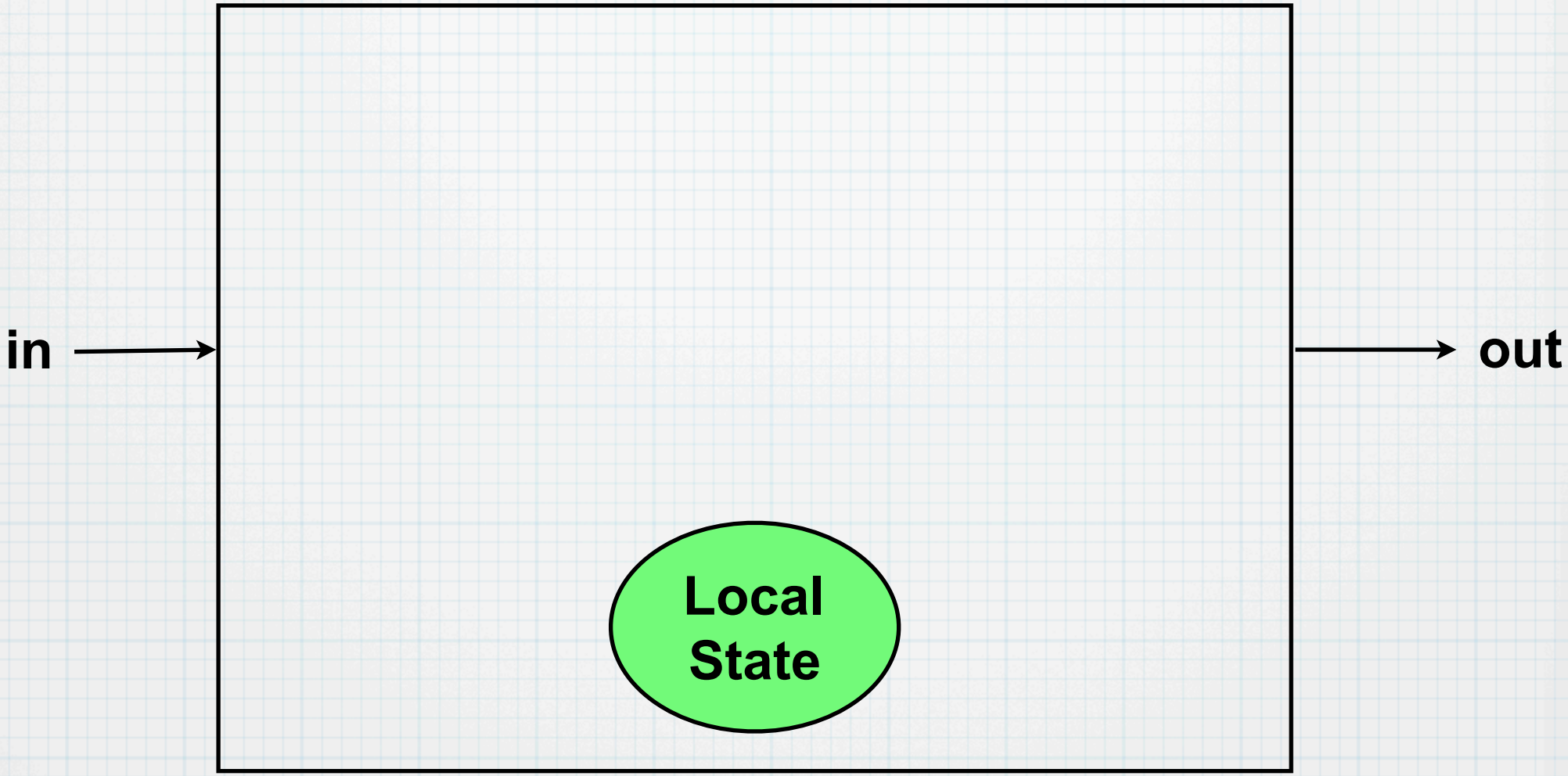
FML-System = (M,con$_d$,con$_a$,input$_d$,input$_a$,output$_d$,output$_a$)

FML - Module

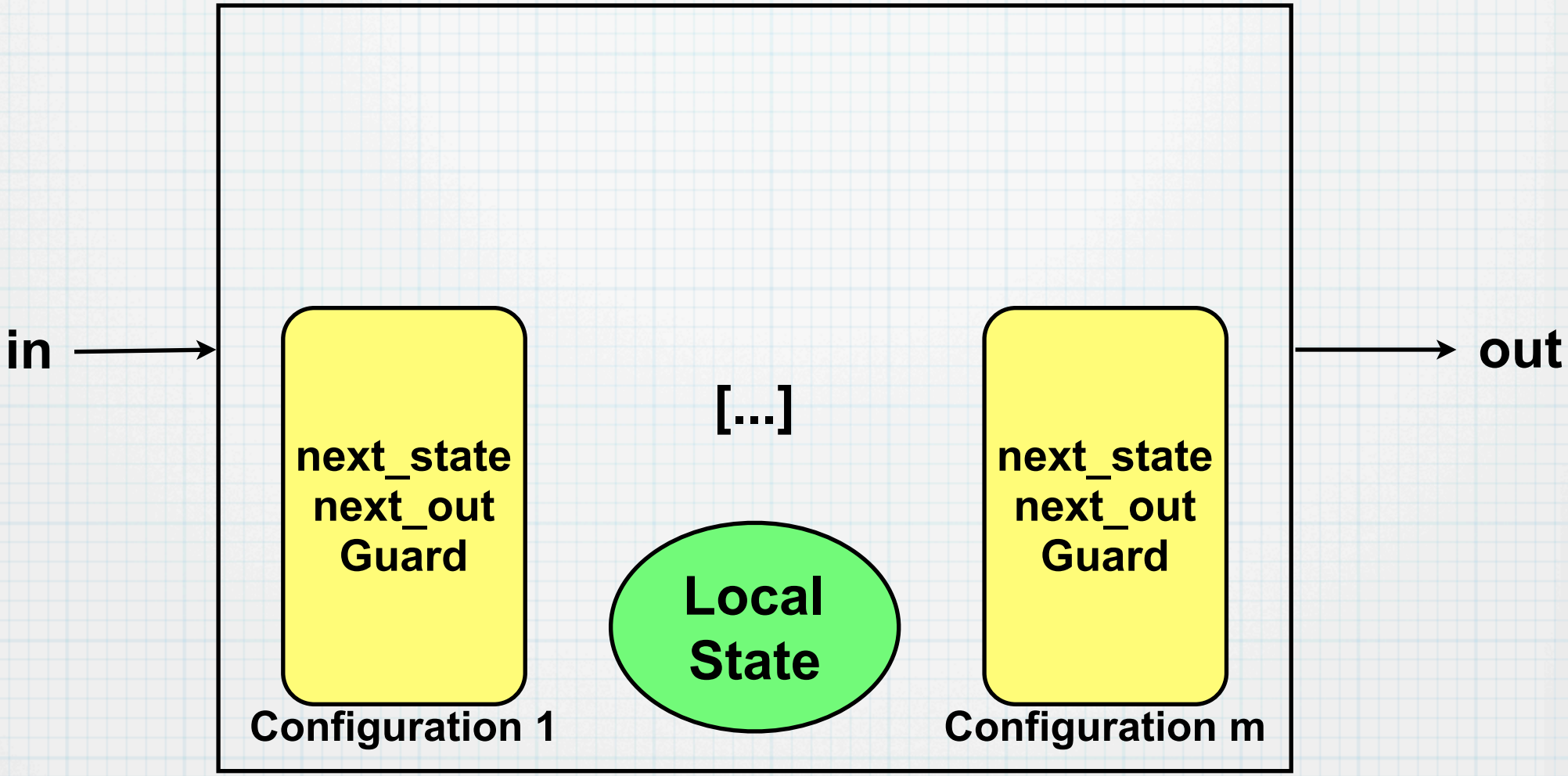
FML - Module



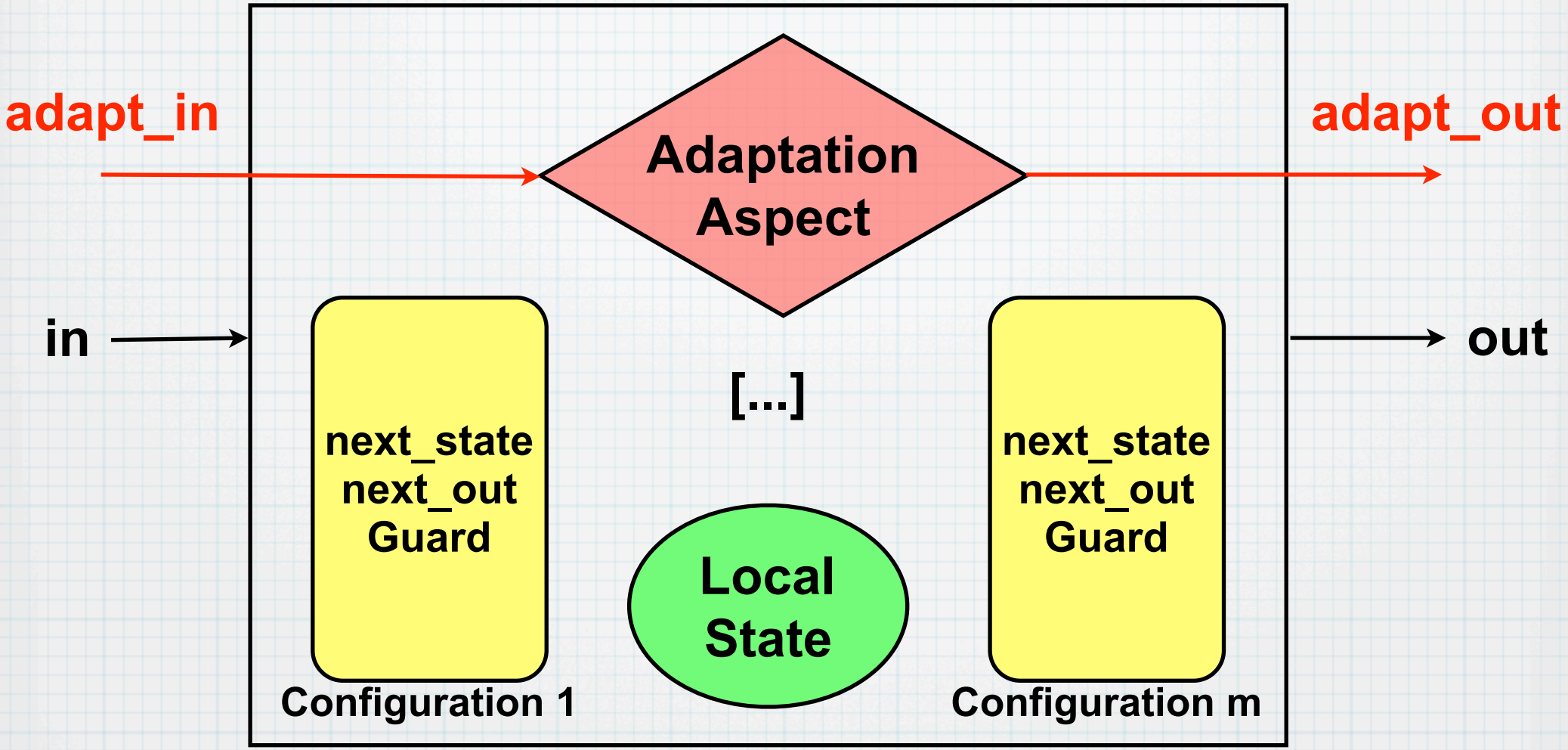
FML - Module



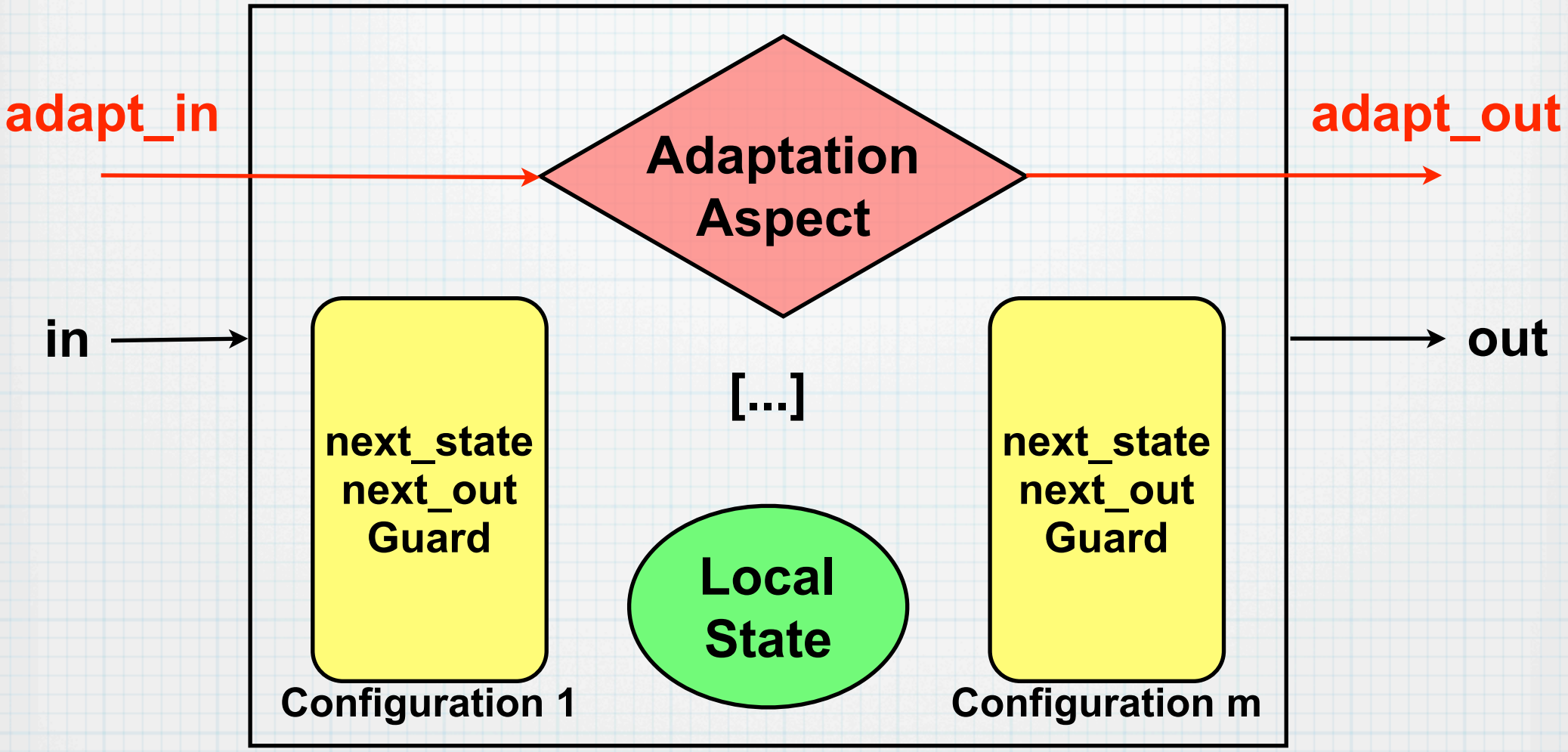
FML - Module


in
next_state
next_out
Guard
[...]
Local
State
next_state
next_out
Guard
out
Configuration 1
Configuration m

FML - Module

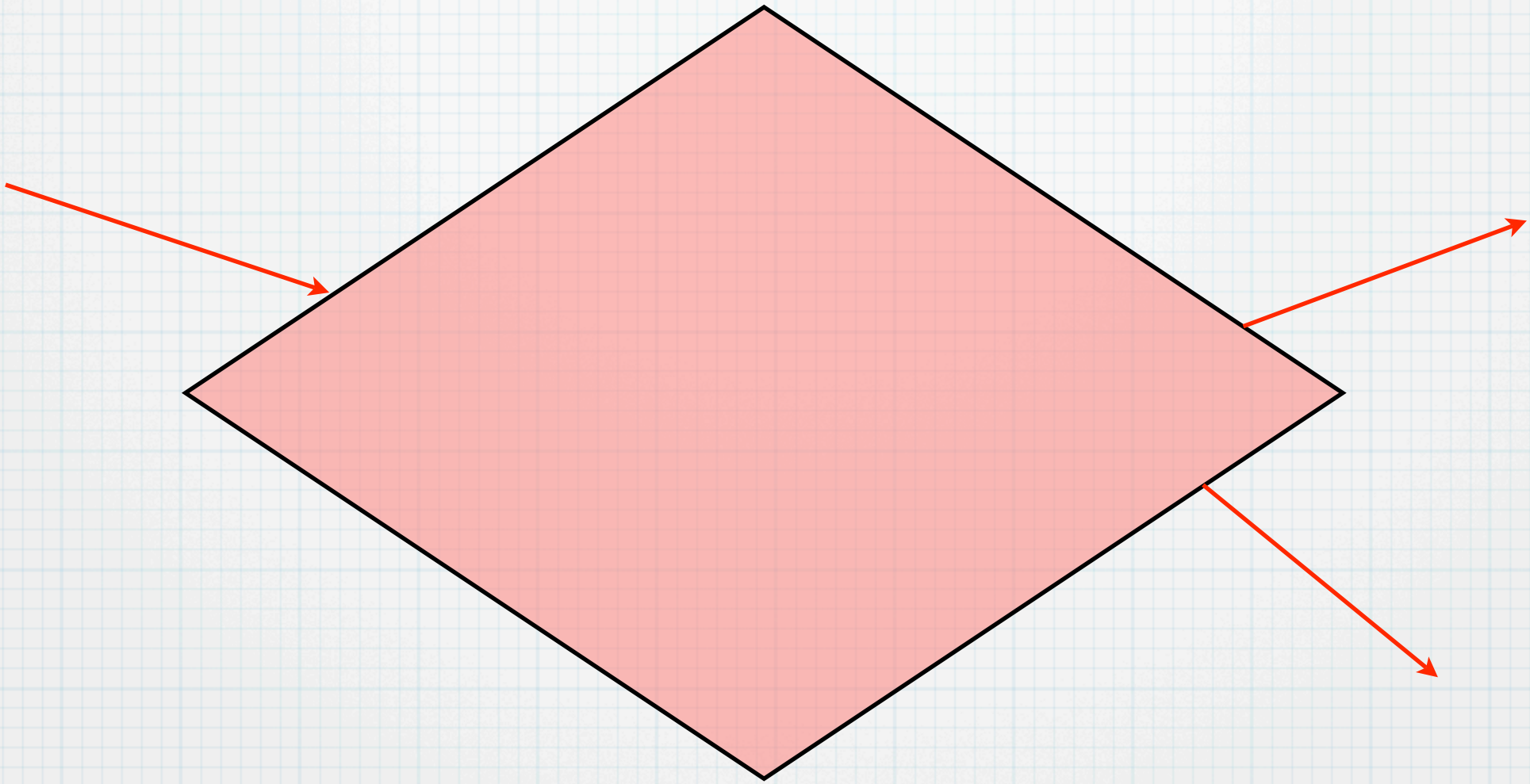

adapt_in
Adaptation Aspect
adapt_out
in
out
next_state
next_out
Guard
[...]
Local State
next_state
next_out
Guard
Configuration 1
Configuration m

FML - Module

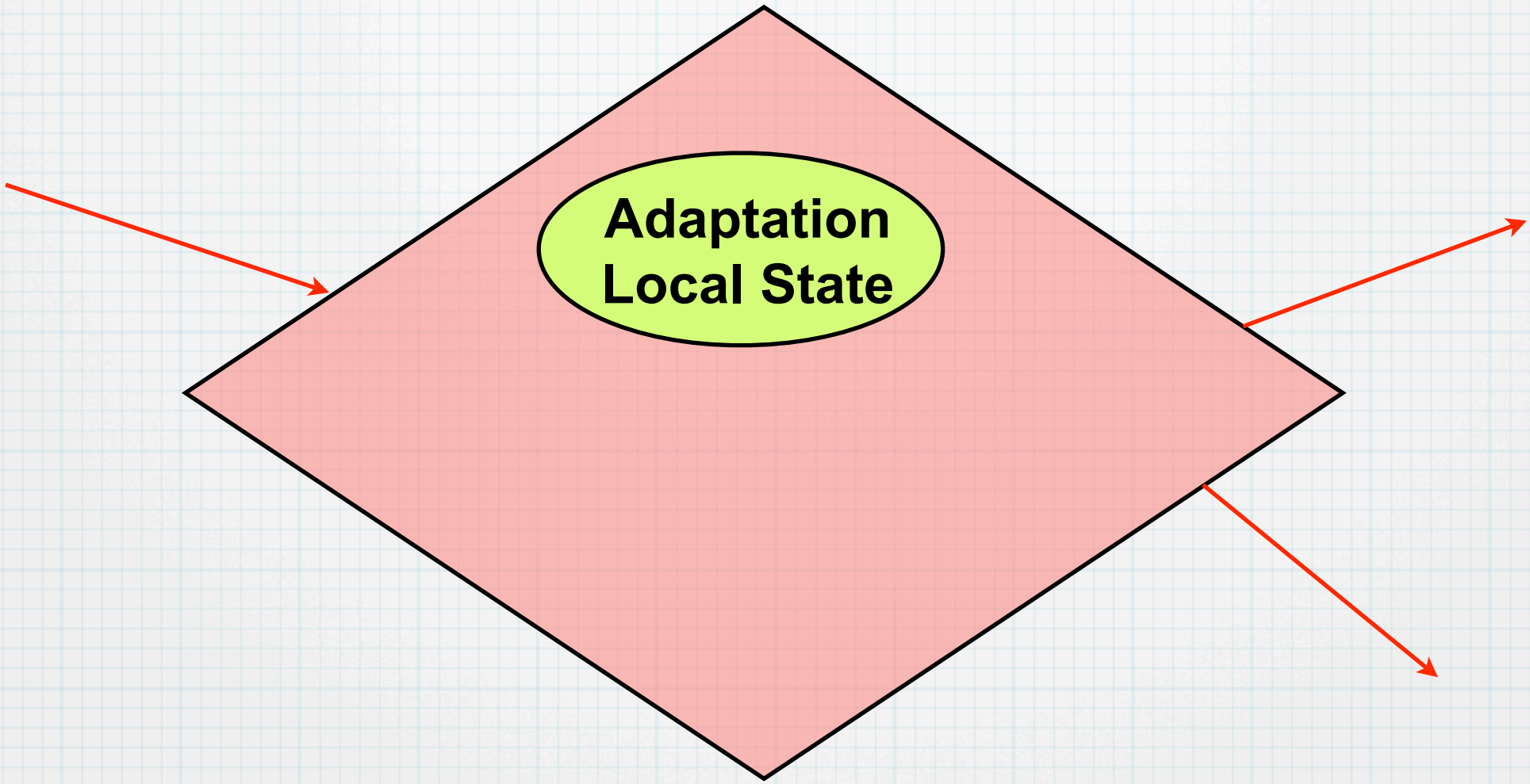


$c_i = (in_i, out_i, loc_i, conf_i, adaptation_i)$

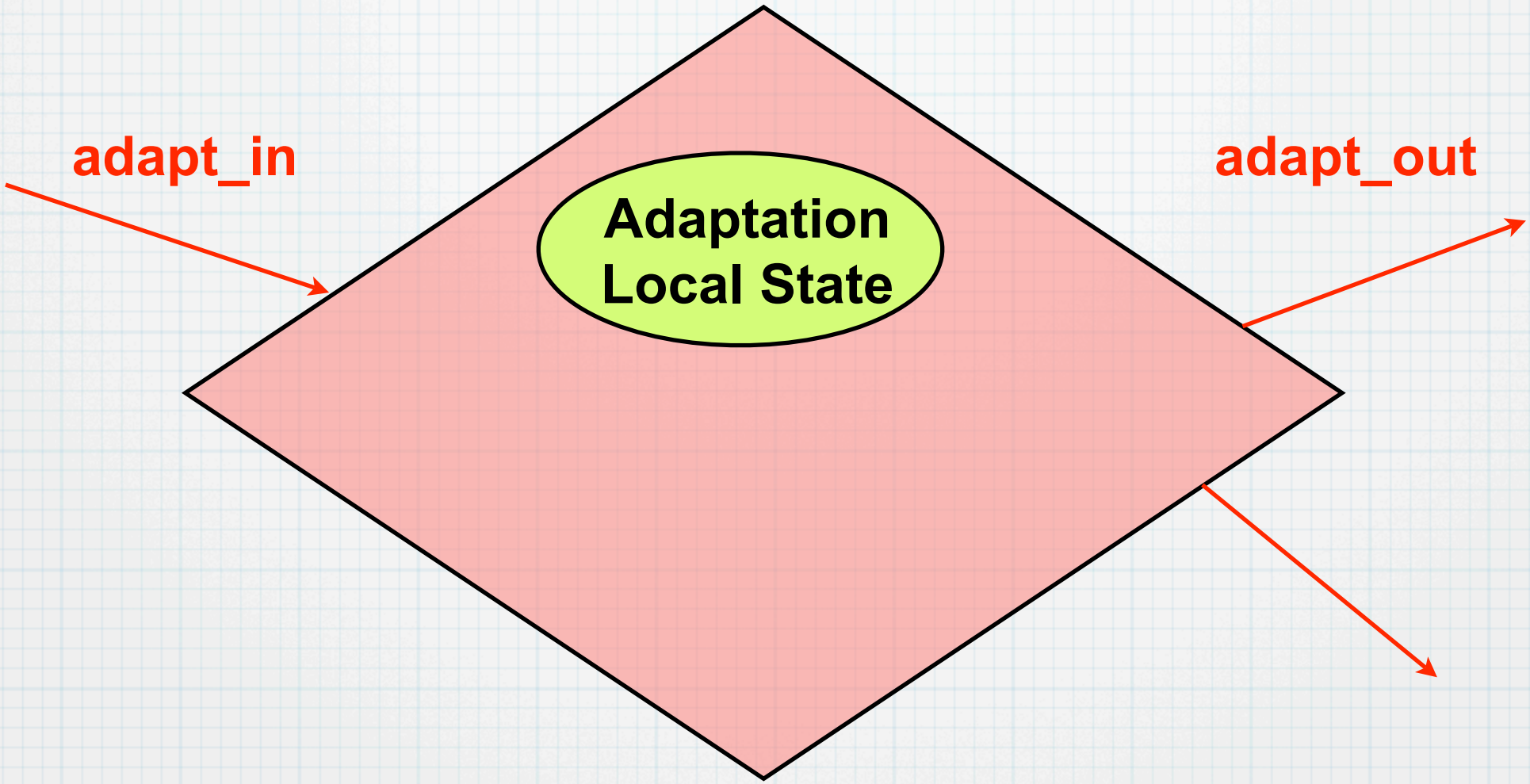
FML Adaptation Aspect



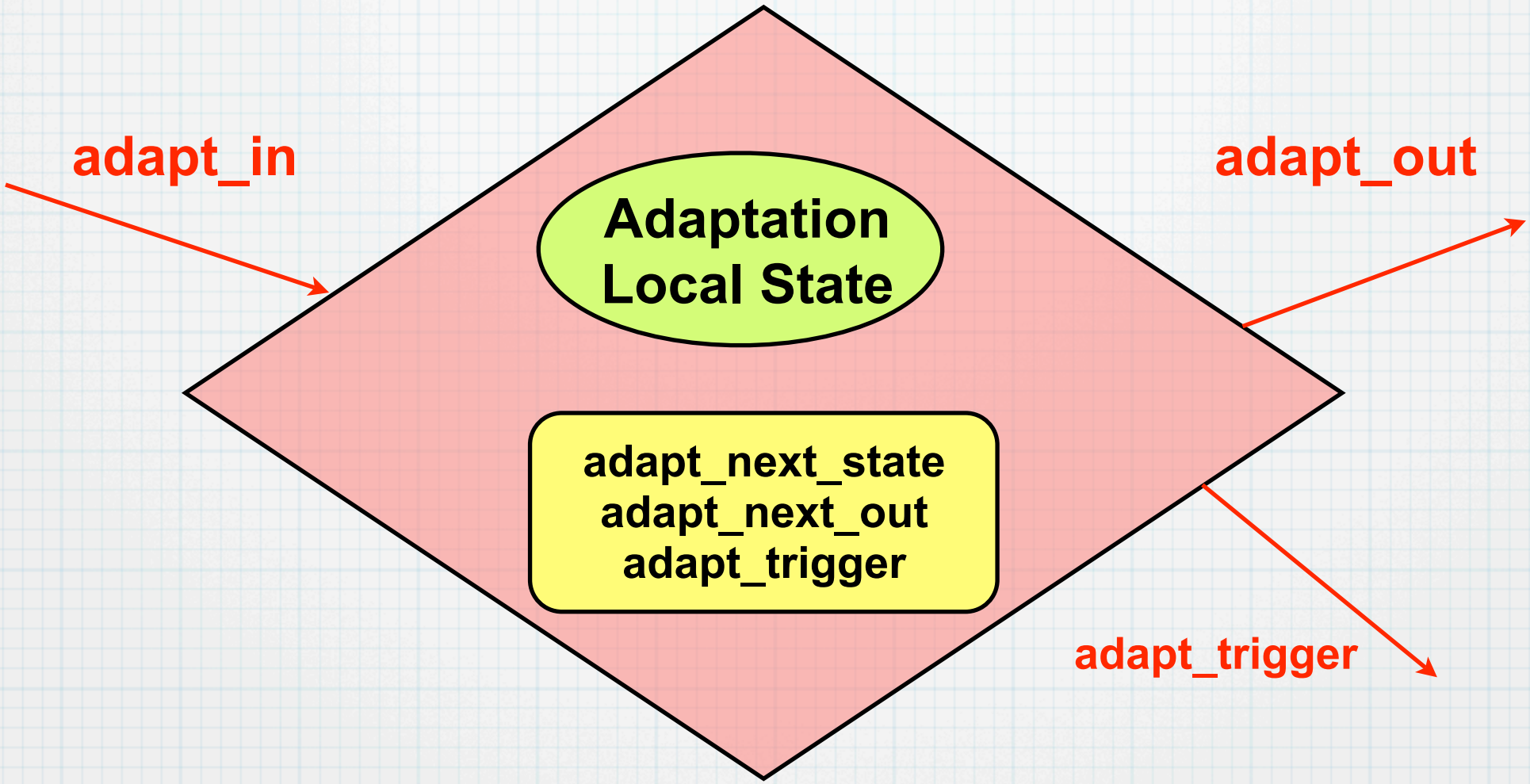
FML Adaptation Aspect



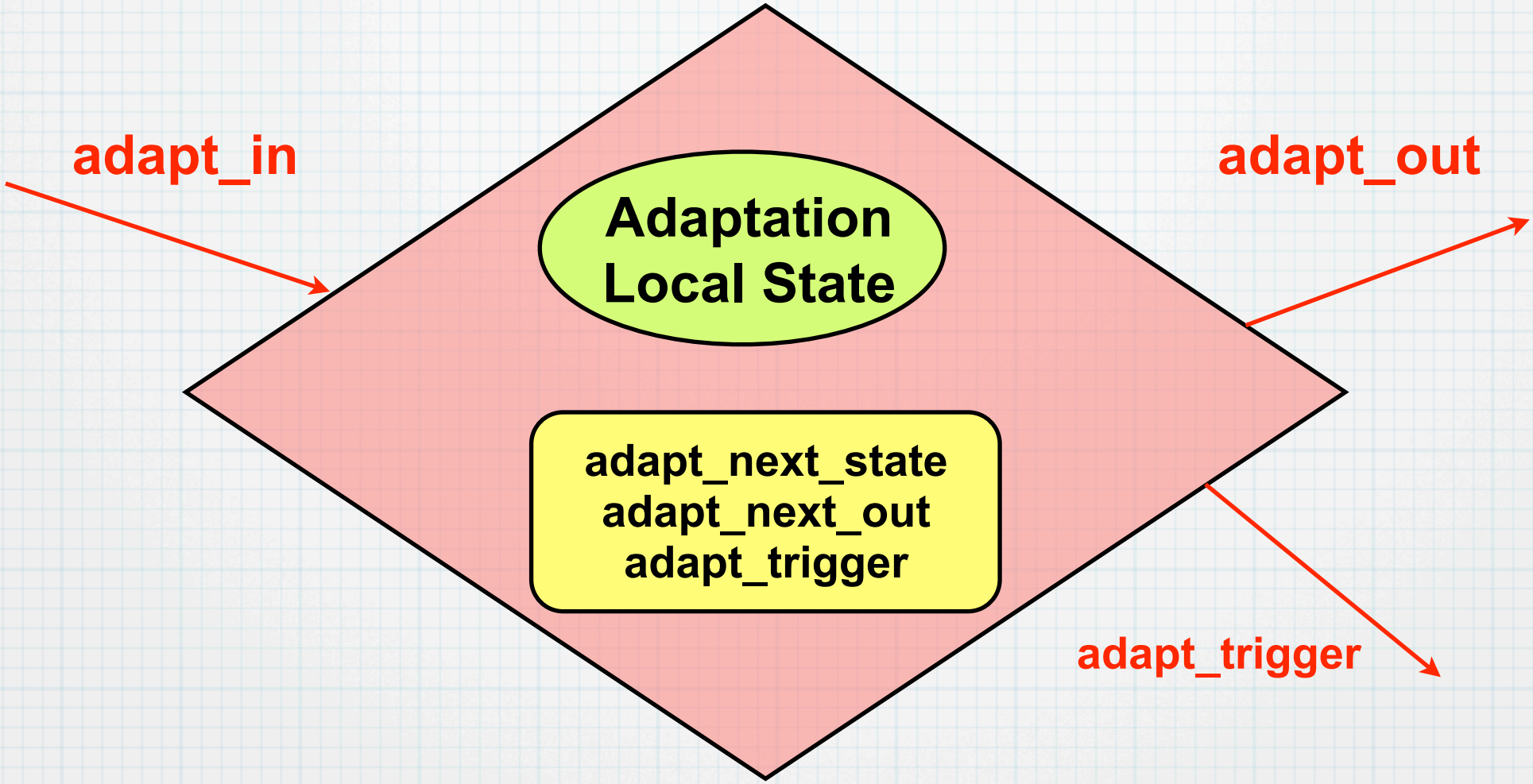
FML Adaptation Aspect



FML Adaptation Aspect



FML Adaptation Aspect



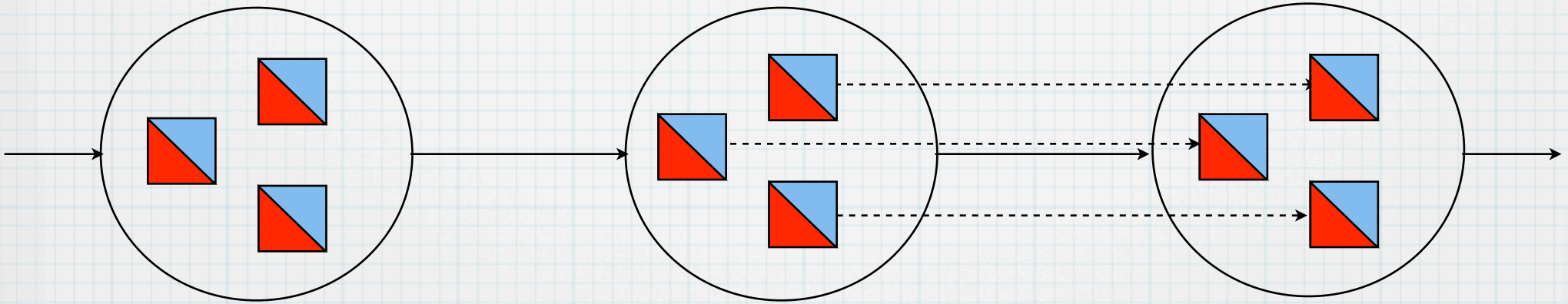
adaptation = (adapt_in, adapt_out, adapt_loc,
adapt_next_state, adapt_next_out, adapt_trigger)

Global FML Semantics

Global States: Set of local states of single modules

$$\sigma = (s_1, s_2, \ldots, s_n)$$

System Traces: Sequences of global states



Global Semantics

$\sigma^i \rightarrow_{glob} \sigma^{i+1}$ is a **global transition** iff

- for all $s_j^i \in \sigma^i$ and for all $s_j^{i+1} \in \sigma_{i+1}$ $s_j^i \rightarrow_{loc} s_j^{i+1}$
- for all k with $con_d(out_k, in_j)$:

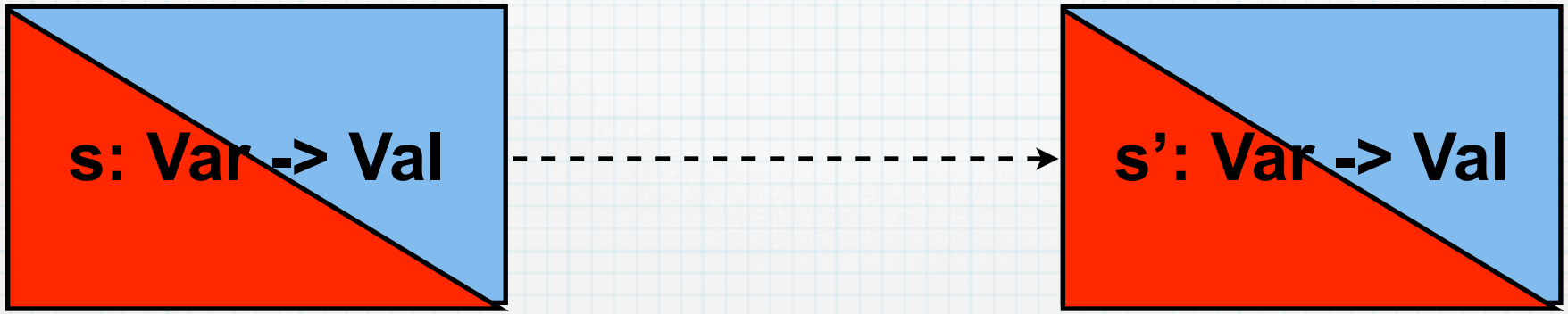
$$s_j^{i+1}|_{in_j \cap out_k} := s_k^{i+1}|_{in_j \cap out_k}$$

- for all k with $con_a(adapt\_out_k, adapt\_in_j)$:

$$s_j^{i+1}|_{adapt\_in_j \cap adapt\_out_k} := s_k^{i+1}|_{adapt\_in_j \cap adapt\_out_k}$$

FML Local Semantics

Local State: Evaluation of in, out and local variables and of adapt_loc, adapt_in and adapt_out variables



Local Transition:

1. Adaptation Aspect determines applicable configuration and computes adaptation parameters.
2. Next_state and next_out function of respective configuration compute next local functional state.

Local Semantics

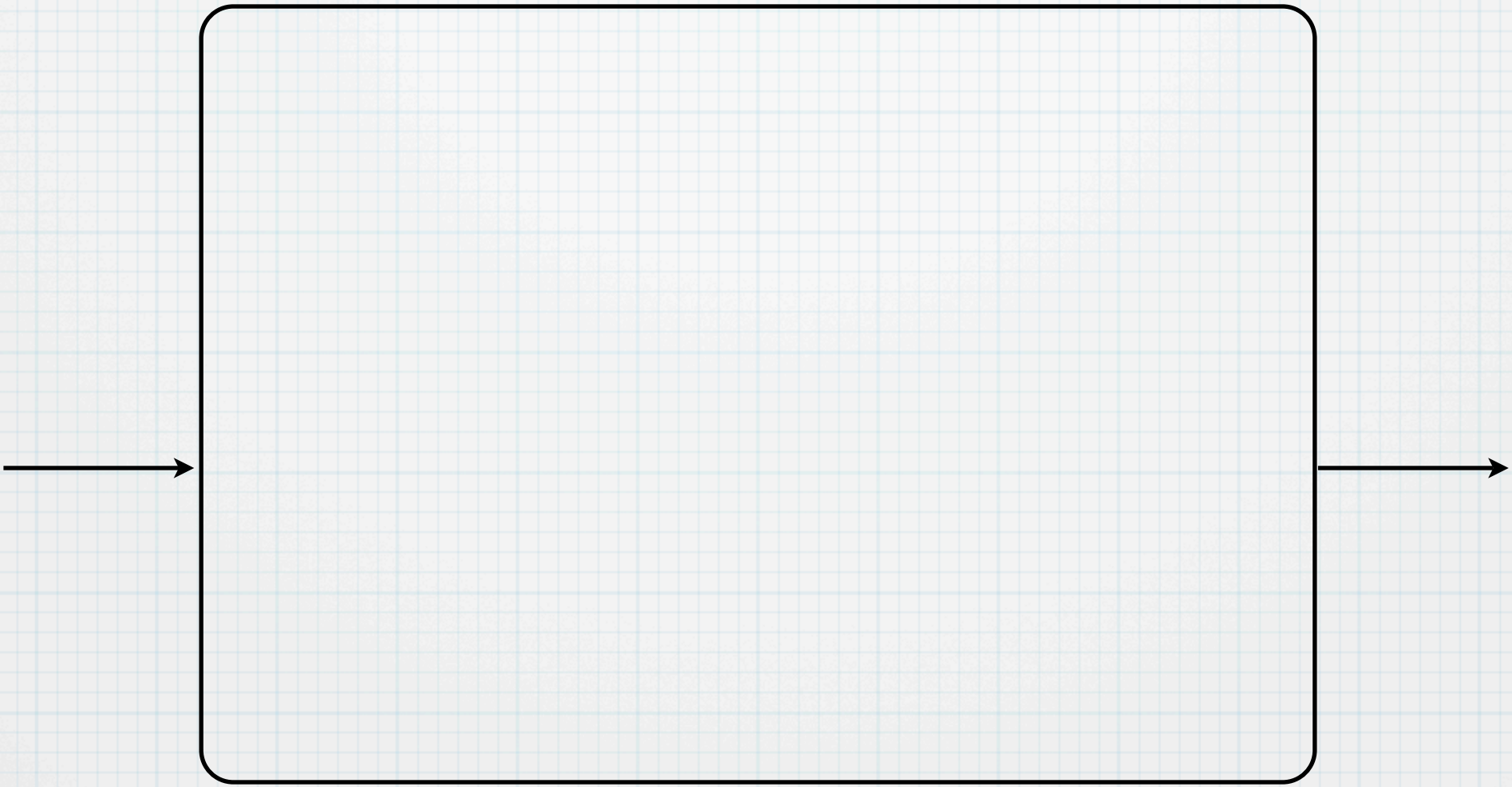
$s \rightarrow_{loc} s'$ is a **local transition** iff

1. $$\begin{aligned} adapt\_trigger[Guards](s) &= i \\ s'|_{adapt\_loc} &= adapt\_next\_state(s) \\ s'|_{adapt\_outs} &= adapt\_next\_out(s) \end{aligned}$$

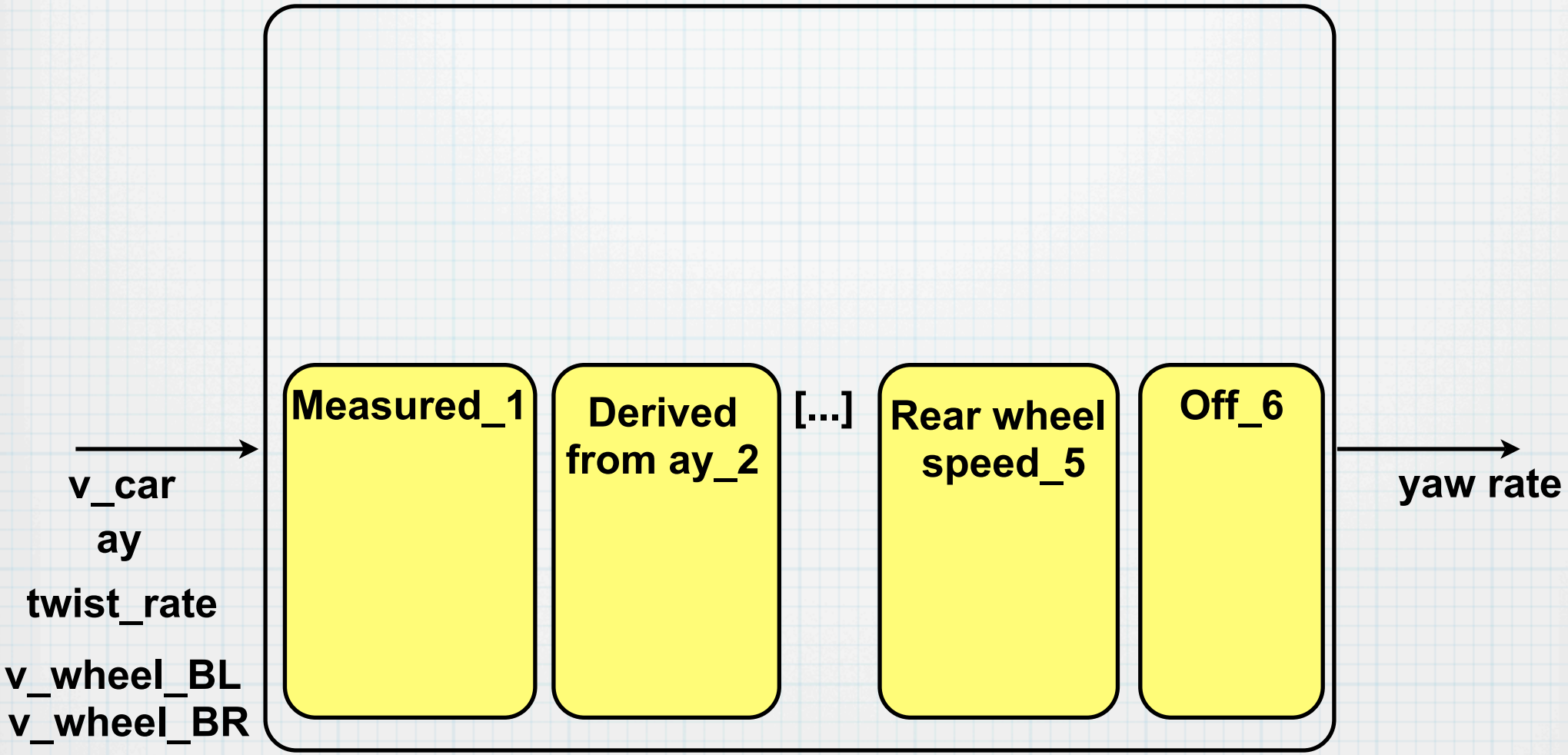
2. $$\begin{aligned} s'|_{loc} &= next\_state_i(s) \\ s'|_{outs} &= next\_out_i(s) \end{aligned}$$

3. $$\begin{aligned} adapt\_trigger[Guards](s) = i \quad &\text{iff} \quad s \models Guard_i \\ \text{and } \forall\ 0 < j < i \quad &\quad\ \ s \not\models Guard_j \end{aligned}$$

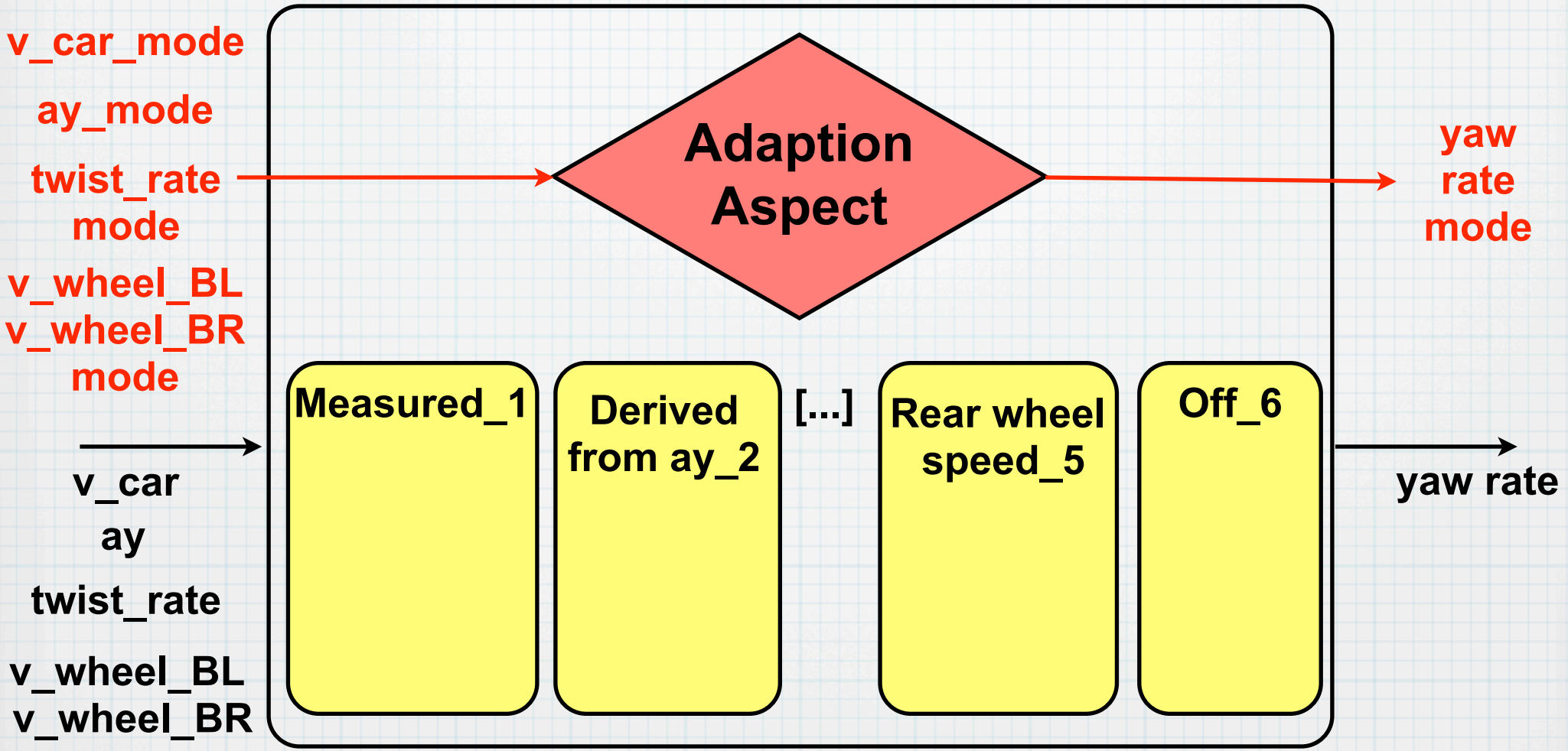
Case Study : CoCar



Case Study : CoCar



Case Study : CoCar



Towards Verification

Classification of System Properties:

- **Functional**
  - In configuration “measured”, no division by zero occurs.
- **Structural-Adaptive**
  - The default configuration is only entered if no other configuration is applicable.
  - All configurations are reachable.
- **Combined**
  - In all reachable configurations, no division by zero occurs.

Towards a Spec Language

On Trace Level:

Evolution of Global states

Temporal operators X, G, F, U, R (cf. LTL)

On System Level:

Modules, Connections, Input, Output

Predicates: module(x), conn(out,in)

On Module Level:

Local Variables, Input, Output, Configurations ...

Predicates: (x = 5), (y > 7), use_conf(3) ...

Towards a Spec Language (2)

Example: All configurations are reachable.

Conf: set of configurations.

use_conf(x): configuration x is used.

$$\forall x \in Conf\,.\ \mathsf{F}\ use\_conf(x)$$

Abstraction

Show: In all configurations, no division by zero occurs.

Measured_1

twist_rate = available

in: twist_rate

yaw_rate := twist_rate

Abstraction

Show: In all configurations, no division by zero occurs.

Measured_1

twist_rate = available

in: twist_rate

yaw_rate := twist_rate

Abstract concrete data domain into finite abstract domain {positive, zero, negative}
Analyse each configuration separately.

Abstraction

Show: In all configurations, no division by zero occurs.

Measured_1

twist_rate = available

in: twist_rate

yaw_rate := twist_rate

Abstract concrete data domain into finite abstract domain {positive, zero, negative}
Analyse each configuration separately.

☑ **No division**

Abstraction (2)

Show: In all configurations, no division by zero occurs.

Rear wheel speed_5

v_wheel_BR = calculated and
v_wheel_BL = calculated

in: v_wheel_BR
in: v_wheel_BL

yaw_rate :=
(v_wheel_BR - v_wheel_BL) / d

where d = track

Abstraction (2)

Show: In all configurations, no division by zero occurs.

Rear wheel speed_5

v_wheel_BR = calculated and
v_wheel_BL = calculated

in: v_wheel_BR
in: v_wheel_BL

yaw_rate :=
(v_wheel_BR - v_wheel_BL) / d

where d = track **positive**

Abstraction (2)

Show: In all configurations, no division by zero occurs.

Rear wheel speed_5

v_wheel_BR = calculated and
v_wheel_BL = calculated

in: v_wheel_BR
in: v_wheel_BL

yaw_rate :=
(v_wheel_BR - v_wheel_BL) / d

where d = track positive

{positive, zero, negative}

Abstraction (2)

Show: In all configurations, no division by zero occurs.

Rear wheel speed_5

v_wheel_BR = calculated and
v_wheel_BL = calculated

in: v_wheel_BR
in: v_wheel_BL

yaw_rate :=
(v_wheel_BR - v_wheel_BL) / d

where d = track positive

☑ **No division by zero, since d is a positive constant.**

{positive, zero, negative}

Abstraction (3)

Show: In all configurations, no division by zero occurs.

Derived from ay_2

(v_car = ax or
v_car = v_wheel)
and ay = measured

in: v_car
in: ay

yaw_rate := ay / v_car

Abstraction (3)

Show: In all configurations, no division by zero occurs.

Derived from ay_2

(v_car = ax or
v_car = v_wheel)
and ay = measured

in: v_car **positive**
in: ay

yaw_rate := ay / v_car

Abstraction (3)

Show: In all configurations, no division by zero occurs.

Derived from ay_2

(v_car = ax or
v_car = v_wheel)
and ay = measured

in: v_car **positive**
in: ay **positive**

yaw_rate := ay / v_car

Abstraction (3)

Show: In all configurations, no division by zero occurs.

Derived from ay_2

(v_car = ax or
v_car = v_wheel)
and ay = measured

in: v_car **positive**
in: ay **positive**

yaw_rate := ay / v_car **positive**

Abstraction (3)

Show: In all configurations, no division by zero occurs.

Derived from ay_2

(v_car = ax or
v_car = v_wheel)
and ay = measured

in: v_car **negative**
in: ay **positive**

yaw_rate := ay / v_car **negative**

Abstraction (3)

Show: In all configurations, no division by zero occurs.

Derived from ay_2

(v_car = ax or
v_car = v_wheel)
and ay = measured

in: v_car **negative**
in: ay **negative**

yaw_rate := ay / v_car **positive**

Abstraction (3)

Show: In all configurations, no division by zero occurs.

Derived from ay_2

(v_car = ax or
v_car = v_wheel)
and ay = measured

in: v_car **zero**
in: ay

yaw_rate := ay / v_car

Abstraction (3)

Show: In all configurations, no division by zero occurs.

Derived from ay_2

(v_car = ax or
v_car = v_wheel)
and ay = measured

in: v_car — zero
in: ay

yaw_rate := ay / v_car

Division by zero !

Abstraction (3)

Show: In all configurations, no division by zero occurs.

Derived from ay_2

(v_car = ax or
v_car = v_wheel)
and ay = measured

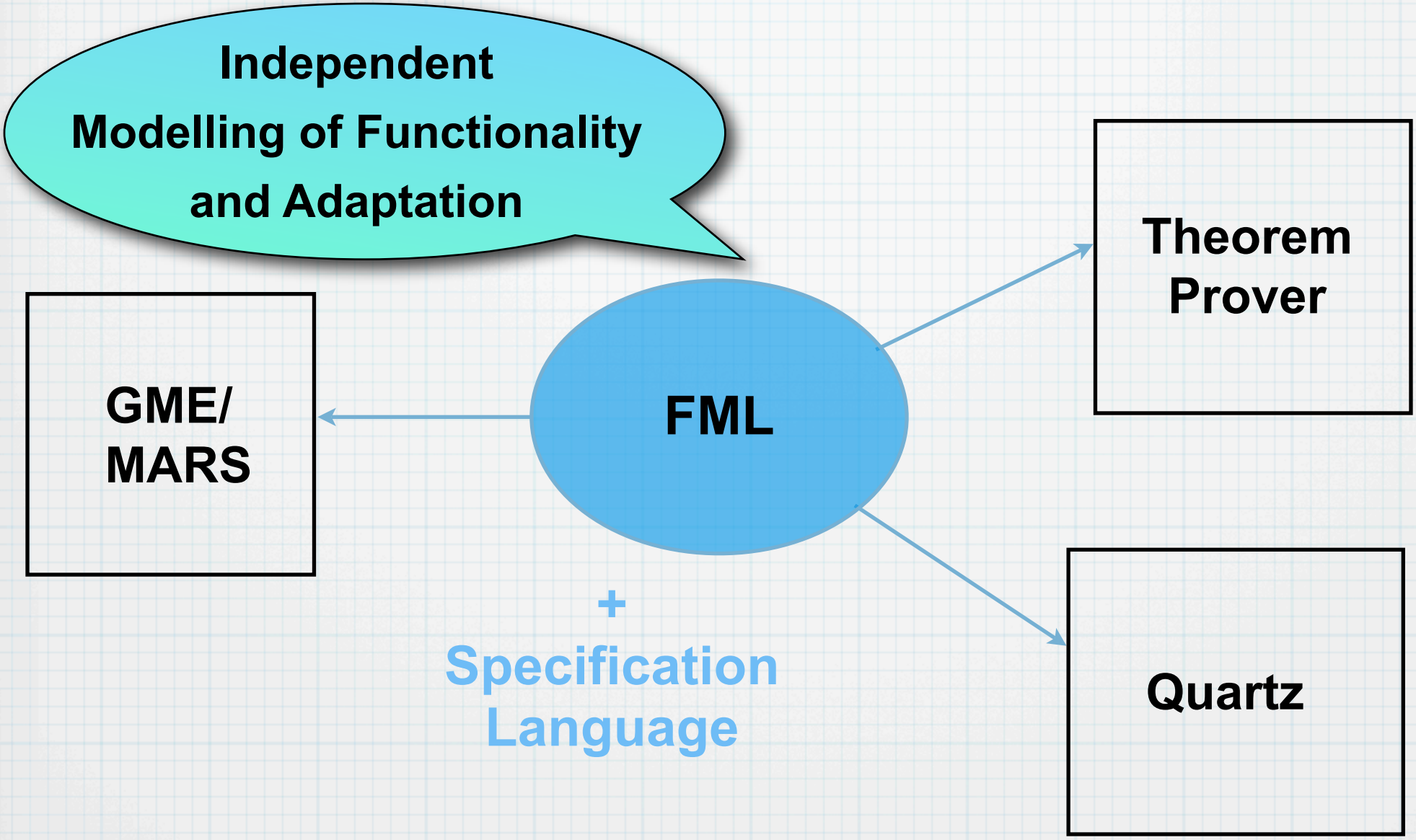
in: v_car **zero**
in: ay

yaw_rate := ay / v_car

Add *(v_car ≠ 0)* to Guard.

Division by zero !

Summary


Independent
Modelling of Functionality
and Adaptation
Theorem
Prover
GME/
MARS
FML
+
Specification
Language
Quartz

Future Work

- Embedding of GME/MARS into FML
- Further Development of Specification Language
- Abstraction Techniques for using QUARTZ
- Integration of Theorem Prover
- Evaluation of Benchmarks