Appendix A

Non-Proprietary Automotive Feature Set: UWFMS

This chapter describes the functionality of each of the automotive feature design models in my non-proprietary set and show how such functionality is modelled using the subset of the STATEFLOW language described in Chapter 4. The set of non-proprietary feature design models is called the "University of Waterloo Feature Model Set" (**UWFMS**). The UWFMS is novel in the sense that there is not a publicly available set of models that I could use to validate my methods to detect feature interactions in the automotive domain. Some of these feature design models are based on TRW Automotive features' textual descriptions provided by their website [5]. These features are regarded by TRW as 'Active Safety Systems', under the heading of 'Driver Assist Systems'. The other features were devised by Richard Fanson[1] and I to have a larger set of feature models to work with.

Each section provides the final STATEFLOW design model per feature used in the case study described in Chapter 7, as well as a brief description of the design decisions made when modelling the UWFMS in STATEFLOW. Each of the non-proprietary feature models were designed to fulfill a goal, with no explicit intention that these features might interact with each other in an unsafe manner. The inputs, outputs and local variables used in each design model are also summarized in a table for each type of variables. These tables are placed close to the design model to help the reader understand the figures.

A.1 Cruise Control (CC)

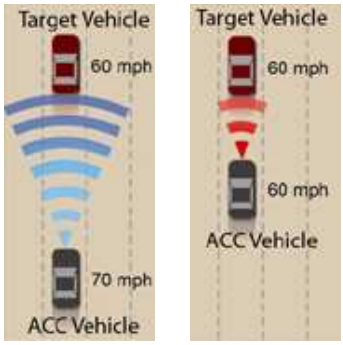
UWFMS's CC is based on TRW's ACC description:

---

[1] Richard Fanson is a Mechatronics engineer who helped in the design of the UWFMS. His knowledge and insights, as well as those from the engineers at GM, helped to make the UWFMS representative [99].

> "TRW's Adaptive Cruise Control (ACC) technology improves upon standard cruise control by automatically adjusting the vehicle speed and distance to that of a target vehicle. ACC uses a long range radar sensor to detect a target vehicle up to 200 meters in front and automatically adjusts the ACC vehicle speed and gap accordingly. ACC decelerates or accelerates the vehicle according to the desired speed and distance settings established by the driver. As per standard cruise control, the driver can override the system at any time".

Figure A.1 shows an example of the feature's execution from the TRW website.



(a) (b)

Figure A.1: Adaptive Cruise Control Functionality: (a) The ACC vehicle approaches the Target vehicle at 70 mph (ACC's vehicle set speed); (b) Due to the proximity, ACC starts coasting by adjusting the ACC vehicle's speed to 60 mph, matching the Target vehicle's speed.

There are two kinds of buttons that can be used when modelling features:

- A *Push button*, which when clicked, causes an action. Given that the triggering of an action is instantaneous, push buttons are normally modelled by events (*e.g.,* Cancel). However, some push buttons can be held. These are modelled as two events, one to capture when it is initially pressed (*i.e.,* SetAccelIn), and another to model the instant at which it is no longer pressed (*i.e.,* SetAccelOut).
- A *Toggle button*, which when clicked, alternates between two states, which are set and unset. Given that the state remains, toggle buttons are normally modelled by data such as a Boolean (*e.g.,* CC_Enabled).

Figure A.2 presents CC's functionality modelled in STATEFLOW, which will be explained in the following paragraphs. Table A.1 shows the local variables, Table A.2 the input variables, and Table A.3 the output variables used in UWFMS's CC design model.

The details of UWFMS's CC design in STATEFLOW are as follows: CC should automatically accelerate and decelerate the CC vehicle based on a target speed specified by the driver and a constant distance separation with a Target vehicle, set at 50 meters. Unlike TRW's CC description, our design uses KPH (kilometers per hour). CC consists of two siblings in an ordered-composition: the LOGIC_CONTROL state, which controls the vehicle logic for the throttle, and the SPEED_SETTING state, which keeps track of the target speed. The default of the LOGIC_CONTROL sibling is the DISABLED state, and CC starts in the DISABLED state when the car is turned on. When the driver turns CC on, by pressing the CC_Enabled button, CC enters the ENABLED state and sets the target speed to 0 (which remains 0 until the feature is engaged).

The default of the ENABLED superstate is the DISENGAGED state. Pressing the Set/Accel button, while driving forward at a speed greater than 40 KPH, makes CC to enter the ENGAGED state. The default of the ENGAGED superstate is the COASTING state. CC must be in the COASTING state when either the current speed exceeds the target speed, or if the vehicle ahead is less than 50 meters away. Coast means that the CC vehicle either maintains or decreases its speed by requesting no throttle, but not by braking. CC moves to the ACCELERATING state when the current speed is less then the target speed and the Target vehicle is farther than 50 meters from the CC vehicle. The throttle output is proportional to the difference between the target speed and the current speed. Depressing the brake or pressing the Cancel button shall cause CC to enter the OVERRIDE state. This state remains active until either the Set/Accel or Resume/Coast buttons is pressed, which makes CC to enter the ENGAGED state. An Error event at any time when CC is on will cause a transition to the FAIL state, and it will not be able to recover from this condition until the CC vehicle is restarted.

The default of the SPEED_SETTING sibling is the HOLD_SPEED state. Pressing the Set/Accel button makes CC to enter the INC_SPEED state and locks in the current speed value as the target speed. If the Set/Accel button is held while the target speed is less than 100 KPH, the target speed is incremented for every cycle unit that occurs. Release of the Set/Accel button sends CC back to the HOLD_SPEED state. Pressing the Resume/Coast button makes CC to enter the DEC_SPEED state and locks in the current speed value as the target speed. If the Resume/Coast button is held while the target speed is greater than 0 KPH, the target speed is decremented for every cycle unit that occurs. Release of the Resume/Coast button sends CC back to the HOLD_SPEED state.

The definition of stable for UWFMS's CC is (sCC=sLOGIC_CONTROL).

Type	**Name**	**Meaning**
data	CC_Engaged	Value that indicates when CC is engaged [Boolean]

Table A.1: Local variables used in Cruise Control (CC)

Type	**Name**	**Meaning**
event	No_event	Signal at a cycle unit for the Stateflow model when no other event is generated
event	SetAccelIn	Signal indicating depression of Set/Accel button
event	SetAccelOut	Signal indicating release of Set/Accel button
event	ResumeCoastIn	Signal indicating depression of Resume/Coast button
event	ResumeCoastOut	Signal indicating release of Resume/Coast button
event	Cancel	Signal indicating depression of Cancel button
event	Error	Signal indicating when an error has occurred
data	CC_Enabled	Driver controlled main power to enable/disable the feature [Boolean]
data	FollowDist	Number to define distance to the vehicle ahead (Value between 0 and 100) [Meters in integers]
data	BrakePedal	Value of brake pedal input represented as a percentage of maximum depression [Percentage in integers]
data	AccelPedal	Value of accelerator pedal input represented as a percentage of maximum depression [Percentage in integers]
data	Speed	Current speed of the vehicle (value within the range of 0 to 100) [KPH in integers]

data	PRNDL_In	The input representing the current gear selection with the following values assumed:
		PRNDL = 0: Park
		PRNDL = 1: Reverse
		PRNDL = 2: Neutral
		PRNDL = 3: Drive
		PRNDL = 4: Low
		[Gear selection in integers]

Table A.2: Input variables used in Cruise Control (CC)

Type	**Name**	**Meaning**
data	set_Throttle	Value proportional to difference between the target speed and current speed [Percentage in integers]
data	TargetSpeed	Target cruising speed of the feature (This variable is an output to display its value) [KPH in integers]

Table A.3: Output variables used in Cruise Control (CC)

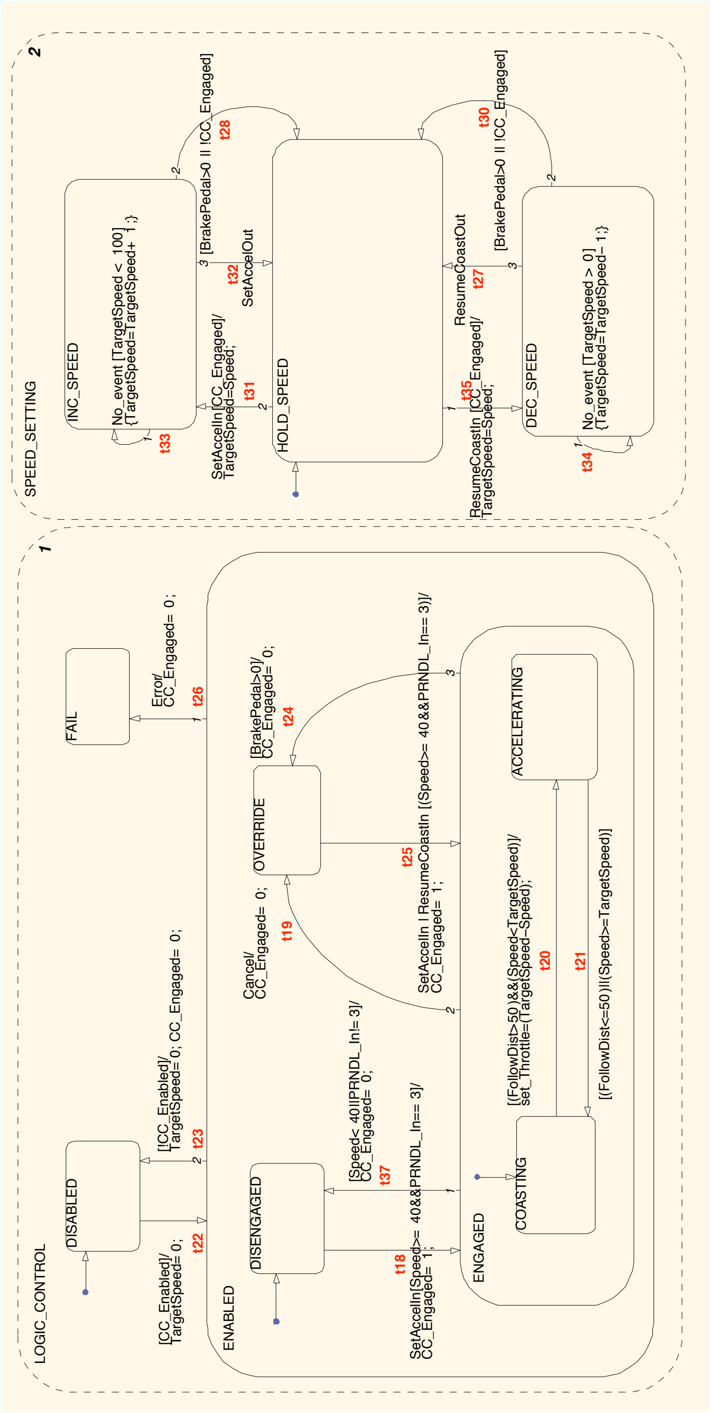

LOGIC_CONTROL
1
DISABLED
FAIL
Error/
CC_Engaged= 0;
t26
[!CC_Enabled]/
TargetSpeed= 0; CC_Engaged= 0;
t23
[CC_Enabled]/
TargetSpeed= 0;
t22
ENABLED
DISENGAGED
OVERRIDE
[BrakePedal>0]/
CC_Engaged= 0;
t24
Cancel/
CC_Engaged= 0;
t19
SetAccelIn | ResumeCoastIn [(Speed>= 40&&PRNDL_In== 3)]/
CC_Engaged= 1;
t25
[Speed< 40||PRNDL_In!= 3]/
CC_Engaged= 0;
t37
SetAccelIn[Speed>= 40&&PRNDL_In== 3]/
CC_Engaged= 1;
t18
ENGAGED
ACCELERATING
COASTING
[(FollowDist>50)&&(Speed<TargetSpeed)]/
set_Throttle=(TargetSpeed-Speed);
t20
[(FollowDist<=50)||(Speed>=TargetSpeed)]
t21
SPEED_SETTING
2
INC_SPEED
No_event [TargetSpeed < 100]
{TargetSpeed=TargetSpeed+ 1;}
t33
[BrakePedal>0 || !CC_Engaged]
t28
SetAccelOut
t32
SetAccelIn [CC_Engaged]/
TargetSpeed=Speed;
t31
HOLD_SPEED
ResumeCoastOut
t27
[BrakePedal>0 || !CC_Engaged]
t30
ResumeCoastIn [CC_Engaged]/
TargetSpeed=Speed;
t35
DEC_SPEED
No_event [TargetSpeed > 0]
{TargetSpeed=TargetSpeed- 1;}
t34

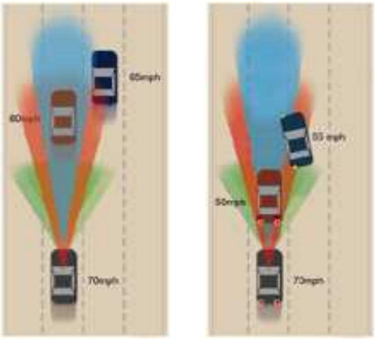
Figure A.2: Cruise Control (CC) STATEFLOW design model

A.2 Collision Avoidance (CA)

UWFMS's CA is based on TRW's CW description:

> "TRW's Collision Warning (CW) System can assist drivers by helping to prevent or mitigate accidents. Combining long and short range radars with a video camera, TRW's Collision Warning monitors the road ahead (including part of the side-fronts). In the event that a vehicle or obstacle approaches, TRW's collision warning system can notify the driver of a possible collision through audible or visual alerts and can also provide braking force as soon as an imminent collision is detected. "

Figure A.3 shows an example of the feature's execution from the TRW website.



(a) (b)

Figure A.3: Collision Warning Functionality: (a) CW uses radars and sensors to monitor the presence and distance of vehicles ahead of the CW vehicle; (b) When CW senses a collision threat with a vehicle in front, it alerts the driver (if a mild threat) and applies brakes (if an imminent threat).

Figure A.4 presents CA's functionality modelled in STATEFLOW, which we will explain in the following paragraphs. UWFMS's CA adds a level of collision threat to the two proposed by TRW's CW. Table A.4 describes the input variables and Table A.5 describes the output variables used in UWFMS's CA design model.

The details of UWFMS's CA design in STATEFLOW are as follows: The CA feature begins in the DISABLED state. When the driver turns CA on by pressing the CA_Enabled button, CA enters the ENABLED state. The default of the ENABLED superstate is the DISENGAGED state. Once the CA vehicle's speed exceeds 25 KPH while driving forward, the feature will enter the ENGAGED state.

When engaged, CA is able to warn the driver and/or take action if a possible threat is encountered. I assumed that some external pre-processing unit computes data from the sensors and converts it into a threat input for CA, which is a reasonable assumption as it occurs in practice in the automotive industry [16]. The threat input could be either: None=0, Mild=1, Near=2, Imminent=3. If a threat input is received while in either the IDLE, WARN, AVOID, or MITIGATE states, one of the following transitions occurs depending on the threat:

- In the case that no threat is detected, or the obstacle ceases to be present, no warning and no braking intervention will be made. CA changes to the IDLE state.
- In the case of a mild threat, a mild threat warning will be output (*i.e.,* $\mathsf{Warning} = 1$) with no braking intervention. CA enters the WARN state.
- In the case of a near threat, a near threat warning will be output (*i.e.,* $\mathsf{Warning} = 2$) and soft braking will occur to slow the vehicle and avoid the potential collision (*i.e.,* $\mathsf{Brake} = 30\%$). CA enters the AVOID state.
- In the case of an imminent threat, an imminent threat warning will be output (*i.e.,* $\mathsf{Warning} = 3$) and full force braking will occur to mitigate the collision as much as possible (*i.e.,* $\mathsf{Brake} = 80\%$). CA enters the MITIGATE state.

If the vehicle is brought to a halt when in either the AVOID or MITIGATE states, CA will go to the HALT state and the vehicle will be held at halt until the driver presses on the brake pedal beyond a threshold (for our model, the brake pedal threshold is set to 10 percent of depression) to resume control of the vehicle. Then, CA will enter the ENABLED but DISENGAGED state. A driver acceleration pedal input exceeding 35 percent of depression will cause CA to enter the OVERRIDE state from any state in the ENABLED superstate. Releasing the acceleration pedal to less than 35 percent of depression while $\mathsf{CA\_Enabled}$ is still true sends CA back to the ENABLED state. An Error event at any time when CA is on will cause a transition to the FAIL state, and it will not be able to recover from this condition until the vehicle is restarted.

UWFMS's CA relies on the fact that the pre-processing threat assessment accounts for vehicle speeds so that what once was a threat will not remain a threat at speeds lower than 25 KPH or at a stop. Otherwise, the feature could be stuck at halt or other undesirable outcomes.

The definition of stable for UWFMS's CA is defined as 1, *i.e., true* because the only one transition is taken in each big-step as ordered-compositions are not present in CA.

Type	**Name**	**Meaning**
event	Error	A signal indicating when an error has occurred
data	CA_Enabled	Driver controlled main power to enable/disable the feature [Boolean]
data	BrakePedal	Value indicating the physical amount of depression of the brake pedal by the driver (0 for not depressed and any positive value when depressed) [Percentage in integers]
data	AccelPedal	Value of physical pedal input represented as a percentage of maximum depression [Percentage in integers]
data	Speed	Current speed of the vehicle (value within the range of 0 to 100) [KPH in integers]

data	Threat	Input from a pre-processing threat assessment block that converts sensor inputs into a threat:
		Threat = 0: No Threat
		Threat = 1: Mild Threat
		Threat = 2: Near Collision Threat
		Threat = 3: Imminent Collision Threat
		[Threat value in integers]

data	PRNDL_In	The input representing the current gear selection with the following values assumed:
		PRNDL = 0: Park
		PRNDL = 1: Reverse
		PRNDL = 2: Neutral
		PRNDL = 3: Drive
		PRNDL = 4: Low
		[Gear selection in integers]

Table A.4: Input variables used in Collision Avoidance (CA)

Type	**Name**	**Meaning**

data	set_Brake	Output indicating if the feature is physically intervening by applying brake force:
		set_Brake = 30: Soft-Braking
		set_Brake = 80: Hard-Braking
		[Brake degree as percentage in integers]

data	CA_HVI	Value to indicate the message being displayed to the driver through a human-vehicle interface (HVI):
		CA_HVI = 0: CA Disabled
		CA_HVI = 1: CA Enabled
		CA_HVI = 2: CA Engaged
		CA_HVI = 3: CA Error
		CA_HVI = 4: CA Override
		[Display value in integers]

data	CA_Warning	Audible and/or visual warning to indicate the presence of threats, their severity and the intervention of CA:
		Warning = 0: No Threat
		Warning = 1: Mild Threat
		Warning = 2: Near Collision Threat
		Warning = 3: Imminent Collision Threat
		Warning = 4: Vehicle Held Stopped
		[Warning value in integers]

Table A.5: Output variables used in Collision Avoidance (CA)

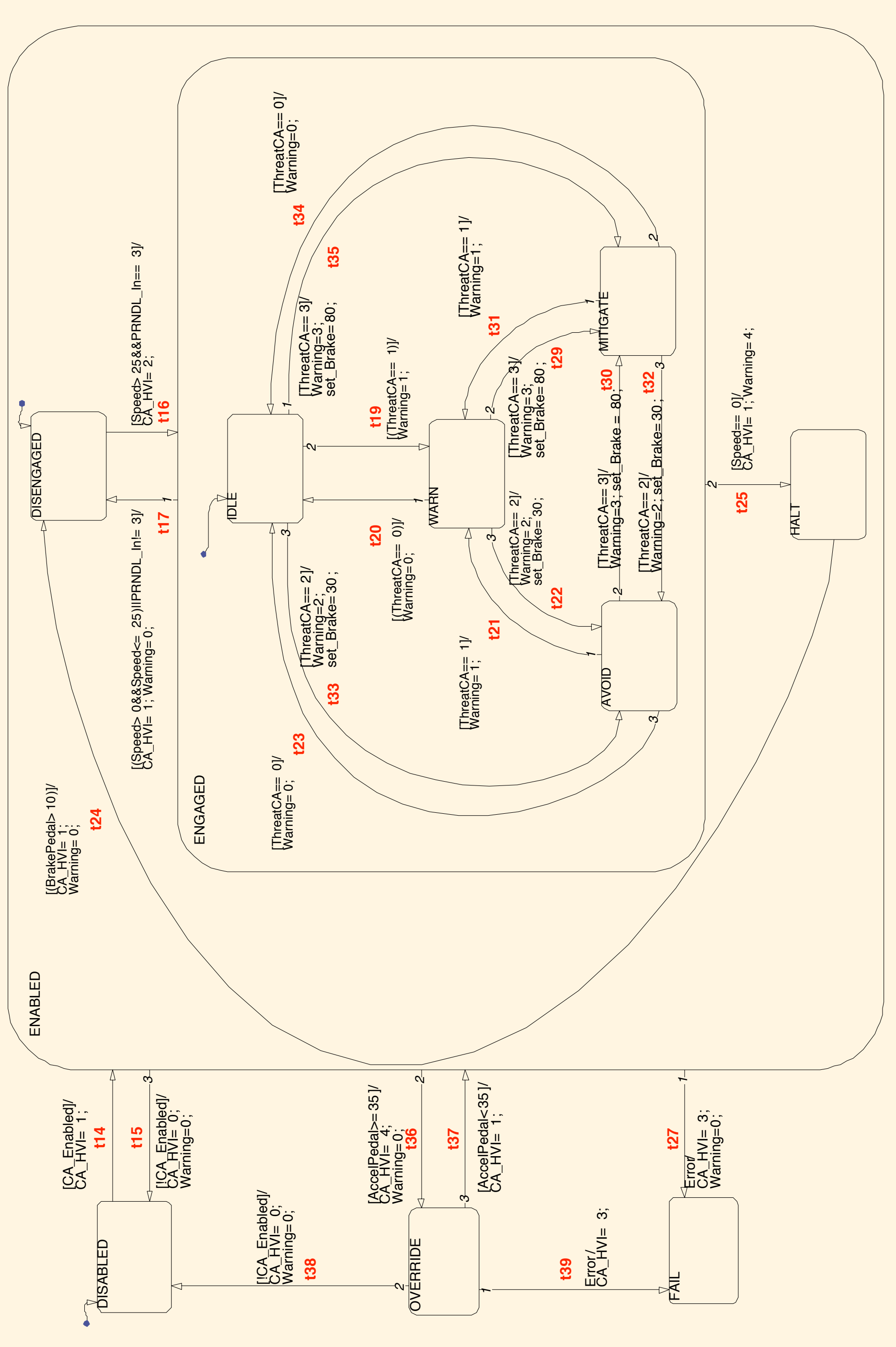

ENABLED
ENGAGED
DISENGAGED
IDLE
WARN
MITIGATE
AVOID
HALT
DISABLED
OVERRIDE
FAIL
t14 [CA_Enabled]/ CA_HVI= 1;
t15 [!CA_Enabled]/ CA_HVI= 0; Warning=0;
t16 [Speed> 25&&PRNDL_In== 3]/ CA_HVI= 2;
t17 [(Speed> 0&&Speed<= 25)||PRNDL_In!= 3]/ CA_HVI= 1; Warning= 0;
t19 [(ThreatCA== 1)]/ Warning=1;
t20 [(ThreatCA== 0)]/ Warning=0;
t21 [ThreatCA== 1]/ Warning= 1;
t22 [ThreatCA== 2]/ Warning= 2; set_Brake= 30;
t23 [ThreatCA== 0]/ Warning= 0;
t24 [(BrakePedal> 10)]/ CA_HVI= 1; Warning= 0;
t25 [Speed== 0]/ CA_HVI= 1; Warning= 4;
t27 Error/ CA_HVI= 3; Warning=0;
t29 [ThreatCA== 3]/ Warning=3; set_Brake= 80 ;
t30 [ThreatCA== 3]/ Warning=3; set_Brake = 80 ;
t31 [ThreatCA== 1]/ Warning=1;
t32 [ThreatCA== 2]/ Warning=2; set_Brake= 30 ;
t33 [ThreatCA== 2]/ Warning=2; set_Brake=30;
t34 [ThreatCA== 0]/ Warning=0;
t35 [ThreatCA== 3]/ Warning=3; set_Brake=80;
t36 [AccelPedal>=35]/ CA_HVI= 4; Warning=0;
t37 [AccelPedal<35]/ CA_HVI= 1;
t38 [!CA_Enabled]/ CA_HVI= 0; Warning=0;
t39 Error/ CA_HVI= 3;
1
2
3

Figure A.4: Collision Avoidance (CA) Stateflow design model

A.3 Park Assist (PA)

UWFMS's PA is based on TRW's PA description:

> "The TRW Park Assist (PA) system combines electrically powered steering with environmental sensing to aid drivers during parallel parking maneuvers. The system uses short range radar sensors to evaluate the length of the parking slot. From this information, the steering trajectory is calculated and the proper steering angle is automatically chosen. The driver monitors the steering. "

Figure A.5 shows an example of the feature's execution from the TRW website.

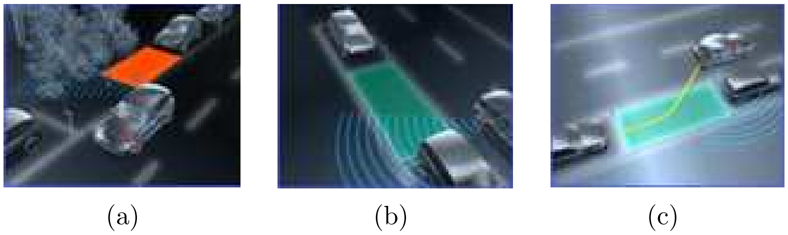

(a) (b) (c)

Figure A.5: Park Assist Functionality: (a) PA monitors for an empty space where the PA vehicle can fit; (b) If a large enough space is found and the driver accepts the space, PA starts the parking maneuver; (c) PA automatically adjusts the steering, throttle and braking during the parking maneuver.

Figure A.6 presents PA's functionality modelled in Stateflow, which we will explain in the following paragraphs. Unlike TRW's PA, in our design, we included the ability for PA to operate the throttle and braking system while the parking maneuver is completed. PA communicates with the driver when some decisions need to be made, such as accepting parking spot, changing gears or enabling the next parking action (*e.g.,* indicate NextPA). Table A.6 describes the input variables and Table A.7 describes the output variables used in the PA's Stateflow design model.

The details of UWFMS's PA design in Stateflow are as follows: PA starts in the DISABLED state when the car is turned on. When the driver turns PA on by pressing the PA_Enabled button, PA enters the ENABLED state. The default of the ENABLED superstate is the IDLE state. The enabled feature PA will remain idle until the speed of the vehicle is less than 10 KPH while driving forward to move to the SEARCHING state and start monitoring the sizes of the adjacent parking spaces. As observed in practice, We assumed that the sensor data is pre-processed by an external module and that PA simply obtains a SpaceFound boolean input indicating when a large enough space has been found. When PA receives the SpaceFound boolean with value true, it will enter the PROMPTING state to prompt the driver (through the HVI) to stop the car and accept the space, or decline the space. If the space is declined or if the driver fails to stop, the external module will make the SpaceFound boolean false, and PA will return to the SEARCHING state to monitor for another parking space. If the driver stopped the car and the space is accepted, PA enters the ENGAGED superstate, entering directly the SWIVEL_OUT state.

While engaged, PA will take over and perform the parallel park maneuver. PA relies on an external component that monitories the progress of the maneuver, sending back to PA a Next event through a HVI when a transition to the next step of the parking maneuver is initiated. PA also relies on the driver to change gears when required, as feature engineers indicated that this process could not be performed automatically by a feature[2]. The maneuver is broken up into 5 steps, each of which correspond to a state:

1. At state SWIVEL_OUT, when receiving a Next event along with variables PRNDL set to 1 (*i.e.,* Reverse) and speed set to a value in the range 1..5, PA will first reverse and turn into the parking space. PA will enter the SWIVEL_IN state.
2. At state SWIVEL_IN, when receiving a Next event along with PRNDL set to 1 (*i.e.,* Reverse) and speed set to a value in the range 1..5, PA will continue to reverse but turn the wheels the other way to swivel the front end into the parking space. PA will move to the STOP1 state.
3. At state STOP1, when receiving a Next event and speed is 0, PA will stop and straighten the wheels. PA will go to the CENTER state.
4. At state CENTER, when receiving a Next event along with PRNDL set to 3 (*i.e.,* Drive) and speed set to a value in the range 1..5, PA will pull forward into the middle of the parking space. PA will enter the STOP2 state.
5. At state STOP2, when speed is 0, PA will finally stop since the maneuver is complete. PA will move to the DISABLED state.

Braking by the driver or the detection of a threat during the maneuver will send PA into the OVERRIDE state, which can be resumed, where the parking process left off, when the threat or braking ceases. Some pre-processing of sensor inputs will compute the sensor data and convert it into a threat input for PA. Steering or acceleration by the driver during the maneuver will send PA to the ABORT state. PA is designed so the feature cannot resume from ABORT since the vehicle is likely off its trajectory path and cannot complete the parking maneuver. An Error event at any time when PA is on will cause a transition to the FAIL mode that cannot be left until the vehicle is shut off and restarted.

The definition of stable for UWFMS's PA is defined as 1, *i.e., true* because the only one transition is taken in each big-step as ordered-compositions are not present in PA.

---

[2] Clarification made by feature design engineers during Alma Juarez's visits to GM Research and Development (2007-2008).

Type	Name	Meaning
event	Next	An signal from a diagnostic external component to indicate when to transition between the different phases of the parking maneuver
event	Error	A signal indicating when an error has occurred
data	PA_Enabled	Driver selected main power to enable/disable the feature [Boolean]
data	SpaceFound	Value to indicate whether or not the sensors have found an appropriately sized space [Boolean]
data	Accepted	Driver's input through a HVI to indicate (when prompted) that the driver accepts the space found, so PA can begin the maneuver [Boolean]
data	Declined	Driver's input through a HVI to indicate (when prompted) that the driver does not accept the space found, so PA looks for another space [Boolean]
data	SteerIn	Driver controlled physical steering wheel input as a steering wheel angle (0 means centred and any other value means steering input from the driver) [Angle in integers]
data	BrakePedal	Value indicating the physical amount of depression of the brake pedal by the driver (0 for not depressed and any positive value when depressed) [Percentage in integers]
data	AccelPedal	Value of physical pedal input represented as a percentage of maximum depression [Percentage in integers]
data	Speed	Current speed of the vehicle (value within the range of 0 to 100) [KPH in integers]
data	Threat	Input from a pre-processing threat assessment block that converts sensor inputs into a threat; For PA, it only indicates presence of obstruction [Boolean]

data	PRNDL_In	The input representing the current gear selection with the following values assumed:
		PRNDL = 0: Park
		PRNDL = 1: Reverse
		PRNDL = 2: Neutral
		PRNDL = 3: Drive
		PRNDL = 4: Low
		[Gear selection in integers]

Table A.6: Input variables used in Park Assist (PA)

Type	**Name**	**Meaning**
data	set_Throttle	Request for throttle control of the vehicle as a percentage of maximum throttle (For PA, only one constant value of throttle percentage is output, which corresponds to a reasonable acceleration for parallel parking) [Percentage in integers]
data	set_Brake	An output request for braking force by the feature as a percentage of maximum braking ability (For PA, only one value of braking is output, which corresponds to soft-braking) [Percentage in integers]
data	set_SteerOut	Output request for steering control of the vehicle (the (value -1 indicates that the vehicle shall turn the wheels to the right, 0 indicates that the wheels shall be centred, and 1 indicates that wheels shall turn to the left. A external component processes these values and manipulate the wheels accordingly) [Steering request in integers]
data	PA_HVI	An output to represent the following information that would be given to the driver: PA_HVI = 0: PA disabled PA_HVI = 1: PA enabled but idle, waiting for speed range to engage searching PA_HVI = 2: PA searching PA_HVI = 3: PA prompting the driver, asking to stop the vehicle and accept or decline the space just found PA_HVI = 4: PA engaged and will display to the driver that it is currently executing the parking maneuver PA_HVI = 5: PA has completed the parking maneuver PA_HVI = 6: PA has had to abort and cannot resume, and the driver will have to try again PA_HVI = 7: PA is being overridden and will continue when reason for override ceases PA_HVI = 8: PA encountered an error and will be unavailable until the vehicle restarts [Display value in integers]

Table A.7: Output variables used in Park Assist (PA)

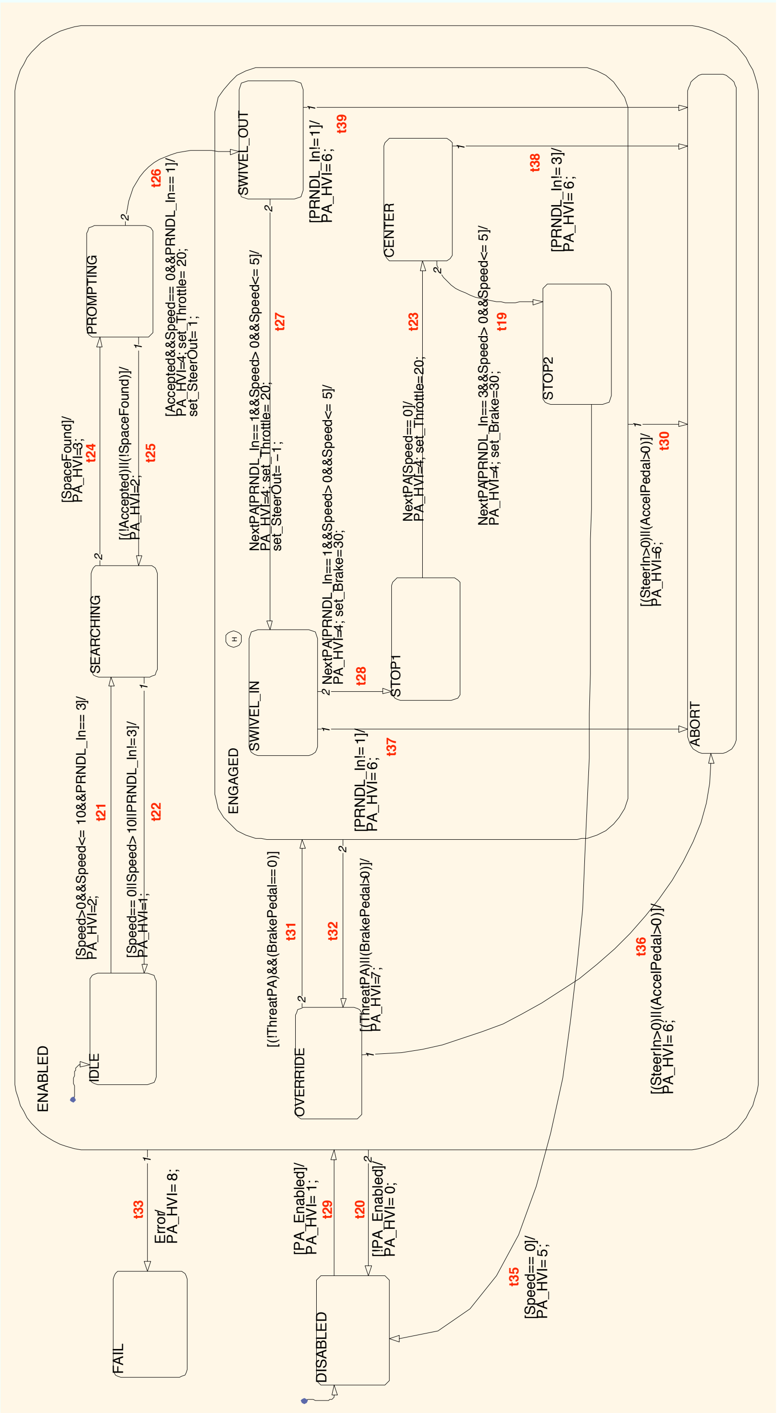


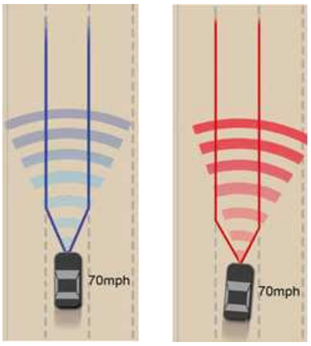
Figure A.6: Park Assist (PA) Stateflow design model

A.4 Lane Guide (LG)

UWFMS's LG is based on TRW's LG description:

> "TRW's Lane Guide Departure (LG) System supports the driver and assists in preventing unintentional lane departures. Utilizing a forward-looking video camera that continuously monitors the vehicle's lane, the system can determine whether or not a driver is unintentionally drifting from the lane or the road. If the driver unintentionally begins to wander out of their lane, the system alerts the driver. "

Figure A.7 shows an example of the feature's execution from the TRW website.



(a) (b)

Figure A.7: Lane Guide Functionality: (a) As the LG vehicle cruises in a lane, LG monitors the lane markings and the vehicle's position; (b) If vehicle drifts from its lane, LG first can warn the driver, and even provide steering corrective input.

Figure A.8 presents LG's functionality modelled in Stateflow, which we will explain in the following paragraphs. Table A.8 describes the input variables and Table A.9 describes the output variables used in the LG's Stateflow design model.

The details of USFMS's LG design in Stateflow are as follows: LG starts in the DISABLED state when the car is turned on. When the driver turns LG on by pressing the LG_Enabled button, LG enters the ENABLED state. The default of the ENABLED superstate is the DISENGAGED state. LG relies on a external module that pre-process the sensor data, which sends back to LG a LaneDrift value indicating the amount of vehicle drifting. If the vehicle is adequately centered in the lane, LaneDrift is equal to 0. If the LaneDrift input reaches the threshold of -10, the vehicle is drifting to the left. If the LaneDrift input reaches the threshold of +10, the vehicle is drifting to the right.

LG can be used in two modes: "Warn" and "Assist". Both modes are contained within the state ENGAGED.

- If LG_Mode = 0, this means that LG is set to "Warn", and the feature will output LG_Warning = 1 to indicate that the vehicle drifts too far to the left or to the right, but no other action other than the warning is taken. While LG_Mode = 0, if the vehicle is drifting left (*i.e.,* LaneDrift < -10), LG will enter the WARN_LEFT state, whereas if the vehicle is drifting right (*i.e.,* LaneDrift > 10), LG will enter the WARN_RIGHT state.
- If LG_Mode = 1, this means that LG is set to "Assist", and the feature will output LG_Warning = 0 (*i.e.,* no warning indication). While LG_Mode = 1, if the vehicle is drifting left (*i.e.,* LaneDrift < -10), LG will enter the ASSIST_LEFT state, and output SteerOut = -1; if the vehicle is drifting right (*i.e.,* LaneDrift > 10), LG will enter the ASSIST_RIGHT state and output SteerOut = 1. These SteerOut output values will indicate some external component to determine and request the steering required to center the car. If the mode is set to "Assist", the warning will not activate when the vehicle drifts because PA will act to center the vehicle in its lane and a warning during the execution of PA's centering would be annoying to the driver.

The turn signal indication, brake pedal depression, and steering wheel input (over a threshold of 10 and -10 degrees of rotation) will all send LG from any state in ENGAGED to the OVERRIDE state. When any of these conditions cease to be present, LG goes to the DISENGAGED state.

An Error event at any time when LG is on will cause a transition to the FAIL state, which will remain the active state until the vehicle is restarted.

The definition of stable for UWFMS's LG is defined as 1, *i.e., true* because the only one transition is taken in each big-step as ordered-compositions are not present in LG.

Type	**Name**	**Meaning**
event	Error	A signal indicating when an error has occurred
data	LG_Enabled	Driver selected main power to enable/disable the feature [Boolean]
data	LG_Mode	Mode selected by the driver, indicating that LG works in either "Warn" mode (LGmode = 0) or "Assist" mode (LGmode = 1) [Mode value in integers]
data	LaneDrift	Value to indicate if the vehicle is centered or not in the lane (input from an external sensor processing component, where a 0 indicates centered, values less than -10 indicate drifting to the left, and values greater than 10 indicates drifting to the right) [Drifting value in integers]
data	SteerIn	Driver controlled physical steering wheel input as a steering wheel angle (For LG, values are within the range (-20,20), with 0 for centered) [Angle in integers]
data	BrakePedal	Value indicating the physical amount of depression of the brake pedal by the driver (0 for not depressed and any positive value when depressed) [Percentage in integers]
data	TurnSignal	Value indicating if the turn signal is on or not [Boolean]

data	PRNDL_In	The input representing the current gear selection with the following values assumed:
		PRNDL = 0: Park
		PRNDL = 1: Reverse
		PRNDL = 2: Neutral
		PRNDL = 3: Drive
		PRNDL = 4: Low
		[Gear selection in integers]

Table A.8: Input variables used in Lane Guide (LG)

Type	**Name**	**Meaning**
data	LG_Warning	Value that indicates if the LG is providing a warning [Boolean]
data	set_SteerOut	An output request for steering control of the vehicle (For LG, the value -1 indicates that the vehicle shall turn the wheels to the right, 0 indicates that the wheels shall be centered, and 1 indicates that wheels shall turn to the left. Some external component will process these values and manipulate the wheels accordingly) [Steer request in integers]

Table A.9: Output variables used in Lane Guide (LG)

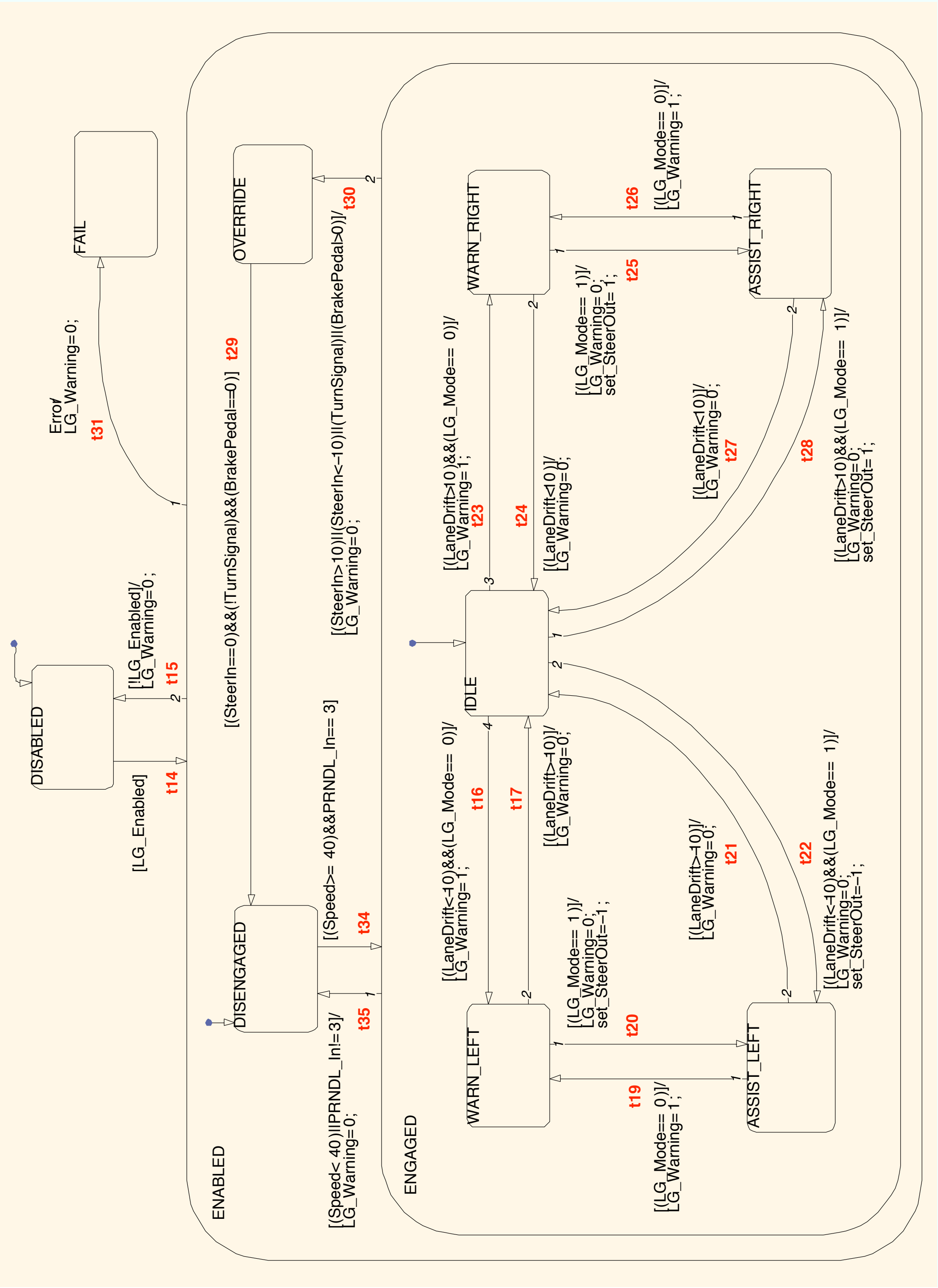


Figure A.8: Lane Guide (LG) Stateflow design model

A.5 Emergency Vehicle Avoidance (EVA)

The Emergency Vehicle Avoidance (EVA) feature assists drivers by pulling over the vehicle in situations when an emergency vehicle, which makes use of a siren, needs the road to be cleared. Combining long and short range radars with a sound detector, and the use of a GPS device, this feature determines when and where the vehicle needs to pull over. When a siren is detected the vehicle should slow and pull to the right-side of the road, and stop if the vehicle is in a safe location. If at an unsafe location (*e.g.,* in the middle of an intersection), the vehicle will coast until it is safe to continue the stop procedure.

Figure A.9 presents EVA's functionality modelled in Stateflow, which we will explain in the following paragraphs. Table A.10 describes the input variables and Table A.11 describes the output variables used in the EVA's Stateflow design model.

The details of UWFMS's EVA design in Stateflow are as follows: EVA starts in the DISABLED state when the car is turned on. When the driver turns EVA on by pressing the EVA_Enabled button, EVA enters the ENABLED state.

At the ENABLED state, the feature will remain idle until a siren from an emergency vehicle is detected, as indicated by the Siren boolean input. When the Siren Boolean is true while driving forward, EVA will enters the ENGAGED state where the feature can acquire control of the vehicle. The default of the ENGAGED superstate is the SLOW state. EVA will monitor the right to see if pulling over is feasible. EVA relies on an external component that sends events back to EVA regarding the safely of a stopping location. Then, EVA receives a WayClear event when the location is safe, or a DontStop event when the location is unsafe. If at the SLOW state EVA gets as an input a WayClear value true, it is safe to pullover farther to the right, and EVA will enter the PULLOVER state. This transition to the PULLOVER state requests braking and also requests steering control by outputting SteerOut = -1, which indicates that the vehicle's wheels need to move to the right. The SteerOut output value will indicate some external component to determine and request the steering required to pull over the vehicle. In either of the SLOW or PULLOVER states, if EVA receives the DontStop boolean input value as true, EVA will enter the COAST state, and also request a slight throttle to keep the vehicle moving until a safe location is found. We assume that the value of throttle requested is adequate to ensure that the vehicle does not come to a stop in an unsafe location (*e.g.,* the middle of an intersection).

Any brake pedal depression or steering input of non-zero, and any acceleration pedal input greater than a 30% depression from the driver sends EVA to the OVERRIDE state from any state in the ENABLED superstate. When any of these conditions cease to be present, EVA goes to the ENGAGED state while the feature is still enabled. Otherwise, EVA transitions to the DISABLED state. An Error event at any time when EVA is on causes a transition to the FAIL state, which remains the active state until the vehicle is restarted.

The definition of `stable` for UWFMS's EVA is defined as 1, *i.e., true* because the only one transition is taken in each big-step as ordered-compositions are not present in EVA.

Type	**Name**	**Meaning**
event	Error	A signal indicating when an error has occurred
data	EVA_Enabled	Driver selected main power to enable/disable the feature [Boolean]
data	Siren	Input from an external component that uses GPS information and microphones to determine if an emergency vehicle is nearby and its whereabouts, so EVA can act by moving the EVA vehicle out of the emergency vehicle's path [Boolean]
data	WayClear	Input from an external component that uses GPS and radar information to indicate if pulling the EVA vehicle over into the far right lane is possible [Boolean]
data	DontStop	Input from an external component that uses GPS information to determine if the current braking action would cause the EVA vehicle to stop in an unsafe location [Boolean]
data	BrakePedal	Value indicating the physical amount of depression of the brake pedal by the driver (0 for not depressed and any positive value when depressed) [Percentage in integers]
data	AccelPedal	Value of physical pedal input represented as a percentage of maximum depression [Percentage in integers]
data	Speed	Current speed of the vehicle (value within the range of 0 to 100) [KPH in integers]

data	PRNDL_In	The input representing the current gear selection with the following values assumed:
		PRNDL = 0: Park
		PRNDL = 1: Reverse
		PRNDL = 2: Neutral
		PRNDL = 3: Drive
		PRNDL = 4: Low
		[Gear selection in integers]

Table A.10: Input variables used in Emergency Vehicle Avoidance (EVA)

Type	**Name**	**Meaning**
data	set_Throttle	Request for throttle control of the vehicle as a percentage of maximum throttle (For EVA, only one constant value of throttle percentage is output, which corresponds to a reasonable acceleration for pulling over) [Percentage in integers]
data	set_Brake	An output request for braking force by the feature as a percentage of maximum braking ability (For EVA, only two values of braking are output, corresponding to soft-braking and mid-force braking) [Percentage in integers]
data	set_SteerOut	An output request for steering control of the vehicle (For EVA, the value -1 indicates that the vehicle shall turn the wheels to the right, 0 indicates that the wheels shall be centred, and 1 indicates that wheels shall turn to the left. Some external component will process these values and manipulate the wheels accordingly) [Steer request in integers]
data	EVA_HVI	An output to represent the following information that would be given to the driver: EVA_HVI = 0: EVA disabled EVA_HVI = 1: EVA enabled, but idle waiting for the an emergency vehicle to be detected (indicated by the siren boolean) EVA_HVI = 2: EVA engaged and executing evasive action to slow and get as far out of the way as possible EVA_HVI = 3: EVA is being overridden EVA_HVI = 4: EVA has encountered an error and will not be available again until the vehicle is restarted [Display value in integers]

Table A.11: Output variables used in Emergency Vehicle Avoidance (EVA)

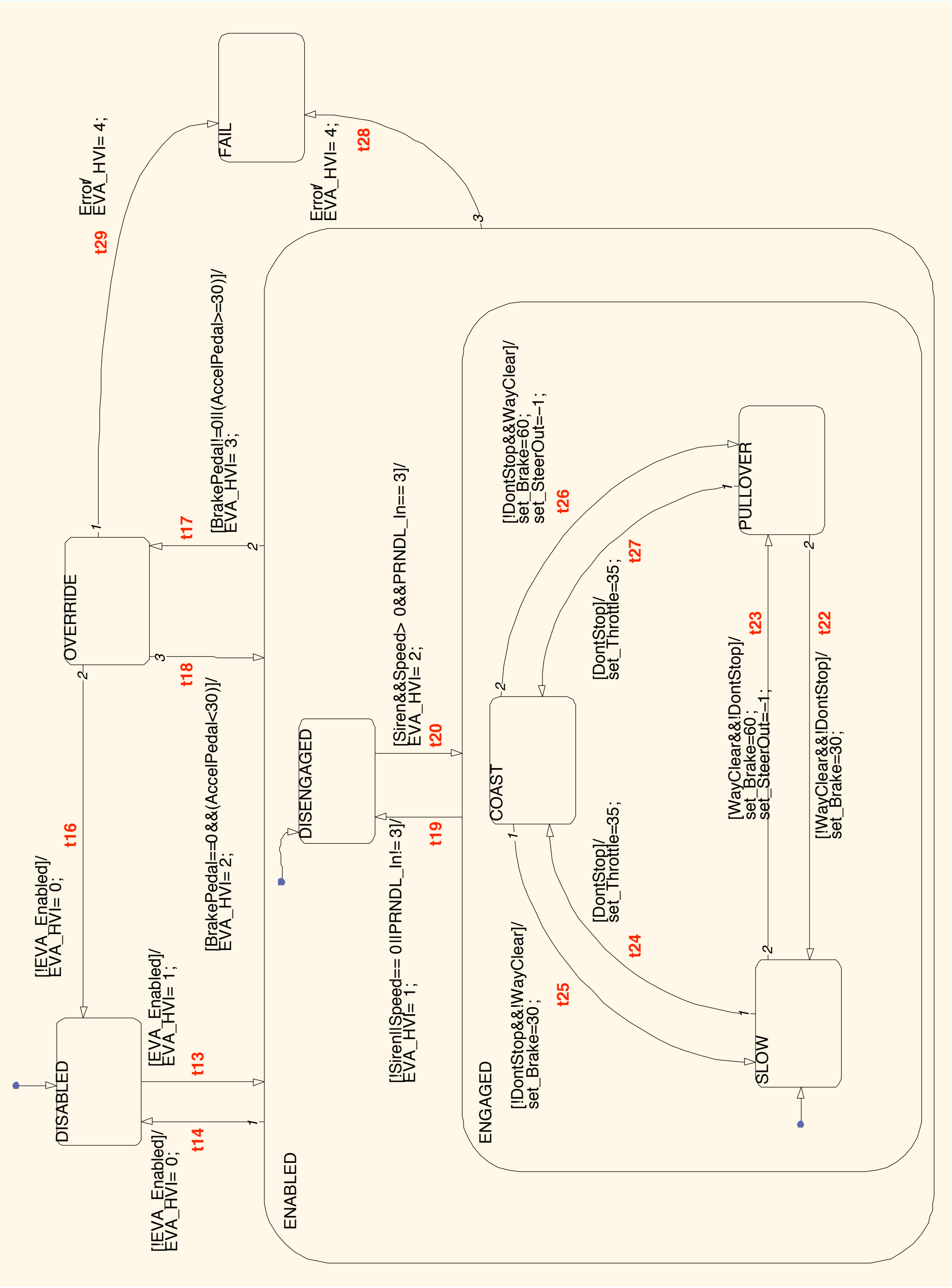


Figure A.9: Emergency Vehicle Avoidance Stateflow design model

A.6 Parking Space Centering (PSC)

The Parking Space Centering (PSC) feature assists drivers during perpendicular parking maneuvers. I assumed that the vehicle is already at a valid parking space; an error signal is sent if it is detected that this is not the case. The feature uses short range radar sensors to determine the location of the vehicle within the box.

Figure A.10 presents PSC's functionality modelled in STATEFLOW, which we will explain in the following paragraphs. Table A.12 describes the input variables and Table A.13 describes the output variables used in the PSC's STATEFLOW design model.

The details of UWFMS's PSC design in STATEFLOW are as follows: PSC defaults to the DISABLED state when the vehicle is turned on. When the drivers turns PSC on by pressing the PSC_Enabled button, PSC enters the ENABLED superstate, which defaults to the DISENGAGED state. When the vehicle is moving at less than 5 KPH while driving forward, PSC moves into the ENGAGED state, which defaults to the STRAIGHT state. At the STRAIGHT state, the vehicle begins moving straight forward, setting Throttle to 20. We assumed that the inputs for LeftLine, RightLine, and FrontLine are units of distance with 5 being a threshold for close distance to the front, and a threshold for not being centered. If the vehicle is farther from the left side than the right side (*i.e.,* LeftLine-RightLine > 5) then the feature enters the MOVE_LEFT state to correct the vehicle's position by setting SteerOut to -1 while Throttle $= 20$. If the vehicle is farther from the right side than the left side (*i.e.,* RightLine-LeftLine > 5) then the feature enters the MOVE_RIGHT state to correct the vehicle's position by setting SteerOut to 1 while Throttle $= 20$. These SteerOut output values will indicate some external component to determine and request the steering required to center the car, with the value of -1 used to represent the vehicle moving its wheels to the left and the value of +1 used to represent wheels pointed to the right. At any point while at the ENABLED state, when the front line of the parking box is 5 units with the difference between the two sides being less than 5, PA enters the HALT state. When the vehicle stops completely, PSC moves to the DISENGAGED state.

Any Error event at any time when PSC is on will cause the feature to transition to the FAILED state, which cannot be left until the vehicle is shut off and restarted. Steering, acceleration or braking by the driver sends PSC to the OVERRIDE state. When any of these conditions cease to be present, PSC goes to the ENABLED state while the feature is still enabled. Otherwise, PSC transitions to the DISABLED state.

The definition of stable for UWFMS's PSC is defined as 1, *i.e., true* because the only one transition is taken in each big-step as ordered-compositions are not present in PSC.

Type	**Name**	**Meaning**
event	Error	A signal indicating when an error has occurred
data	PSC_Enabled	Driver selected main power to enable/disable the feature [Boolean]
data	LeftLine	Input from radar, indicating the distance to the left side of the parking space, whether it is a line marking or an obstacle [Distance in integers]
data	RightLine	Input from radar, indicating the distance to the right side of the parking space, whether it is a line marking or an obstacle [Distance in integers]
data	FrontLine	Input from radar, indicating the distance to the front of the parking space, whether it is a line marking or an obstacle [Distance in integers]
data	BrakePedal	Value indicating the physical amount of depression of the brake pedal by the driver (0 for not depressed and any positive value when depressed) [Percentage in integers]
data	AccelPedal	Value of physical pedal input represented as a percentage of maximum depression [Percentage in integers]

data	PRNDL_In	The input representing the current gear selection with the following values assumed:
		PRNDL = 0: Park
		PRNDL = 1: Reverse
		PRNDL = 2: Neutral
		PRNDL = 3: Drive
		PRNDL = 4: Low
		[Gear selection in integers]

Table A.12: Input variables used in Parking Space Centering (PSC)

Type	Name	Meaning
data	set_Throttle	Request for throttle control of the vehicle as a percentage of maximum throttle (For PSC, only one constant value of throttle percentage is output, which corresponds to a reasonable acceleration for parking) [Percentage in integers]
data	set_Brake	An output request for braking force by the feature as a percentage of maximum braking ability (For PSC, only one value of braking is output, which corresponds to soft-braking) [Percentage in integers]
data	set_SteerOut	An output request for steering control of the vehicle (For PSC, the value -1 indicates that the vehicle shall turn the wheels to the right, 0 indicates that the wheels shall be centred, and 1 indicates that wheels shall turn to the left. Some external component will process these values and manipulate the wheels accordingly) [Steer request in integers]

Table A.13: Output variables used in Parking Space Centering (PSC)

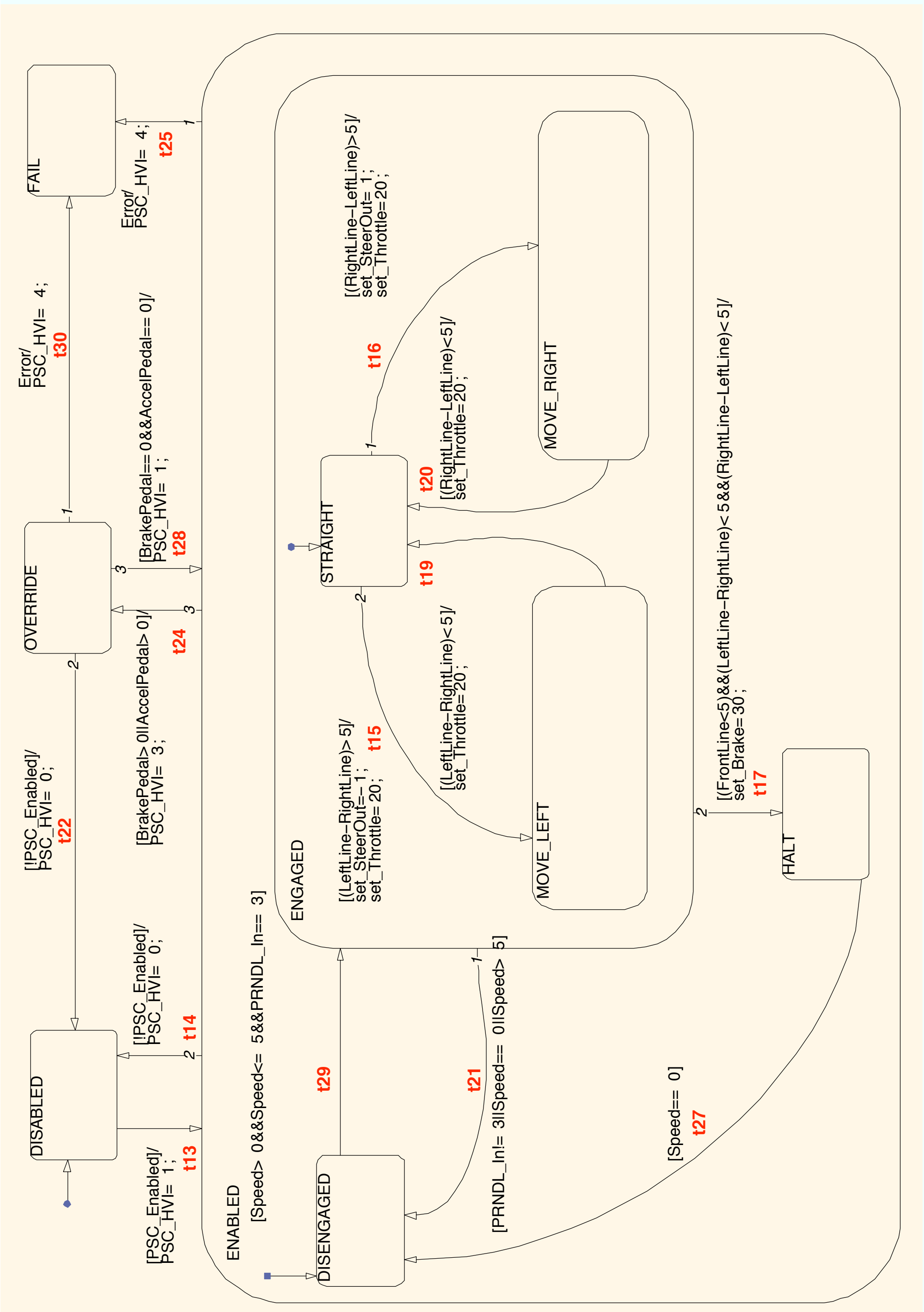

FAIL
OVERRIDE
DISABLED
ENABLED
ENGAGED
STRAIGHT
MOVE_RIGHT
MOVE_LEFT
HALT
DISENGAGED
Error/ PSC_HVI= 4;
t30
Error/ PSC_HVI= 4;
t25
[BrakePedal== 0&&AccelPedal== 0]/ PSC_HVI= 1;
t28
[BrakePedal> 0||AccelPedal> 0]/ PSC_HVI= 3;
t24
[!PSC_Enabled]/ PSC_HVI= 0;
t22
[!PSC_Enabled]/ PSC_HVI= 0;
t14
[PSC_Enabled]/ PSC_HVI= 1;
t13
[Speed> 0&&Speed<= 5&&PRNDL_In== 3]
t29
[PRNDL_In!= 3||Speed== 0||Speed> 5]
t21
[Speed== 0]
t27
[(FrontLine<5)&&(LeftLine−RightLine)< 5&&(RightLine−LeftLine)<5]/ set_Brake=30;
t17
[(RightLine−LeftLine)>5]/ set_SteerOut= 1; set_Throttle=20;
t16
[(RightLine−LeftLine)<5]/ set_Throttle=20;
t20
[(LeftLine−RightLine)>5]/ set_SteerOut=−1; set_Throttle= 20;
t15
[(LeftLine−RightLine)<5]/ set_Throttle=20;
t19

Figure A.10: Parking Space Centering (PSC) Stateflow design model

A.7 Reversing Assistance (RA)

The Reversing Assistant (RA) feature can assist drivers by helping to prevent or mitigate collisions while reversing. Combining long and short range radars, RA monitors the path of the vehicle while reversing. In the event that a vehicle or obstacle approaches, RA notifies the driver of a possible collision and also brakes as soon as the threat of a collision becomes imminent.

Figure A.11 presents RA's functionality modelled in STATEFLOW, which we will explain in the following paragraphs. Table A.14 describes the input variables and Table A.15 describes the output variables used in the RA's STATEFLOW design model.

The details of UWFMS's RA design in STATEFLOW are as follows: RA will start at the DISABLED state when the vehicle is turned on. RA turns on by toggling the RA_Enabled button, making RA to enter the ENABLED superstate, which defaults to DISENGAGED. RA will remain in the DISENGAGED state until the vehicle is put into reverse, indicated by the input PRNDL_In = 1, while the vehicle moves between 10 and 25 KPH. This action will cause RA to enter the ENGAGED superstate, which defaults to the IDLE state. When engaged, RA will be able to warn the driver and/or take action if a possible threat is encountered. RA relies on an external component to pre-process sensor inputs and convert them into a threat input for RA, stored in ObstacleZone. The threat input data could be either: None=0, Mild=1, Imminent=2. If a threat input is received while in either the IDLE, WARN, or ASSIST states, one of the following transitions will occur depending on the threat:

- In the case that no threat is detected, or the obstacle ceases to be present, no warning and no braking intervention are made. RA changes to the IDLE state.
- In the case of a mild threat, a mild threat warning will be output (*i.e.,* Warning = 1) with no braking intervention. RA enters the WARN state.
- In the case of an imminent threat, an imminent threat warning will be output (*i.e.,* Warning = 2) and a braking request will be made to mitigate the collision as much as possible (*i.e.,* Brake = 60%). RA enters the ASSIST state.

If the vehicle stops while the feature is in the ASSIST state, RA will enter the HOLD state and remain there until an input of BrakePedal $>$ 20 is detected, so RA moves to the DISENGAGED state and the driver can regain control of the vehicle. Whenever the vehicle's gear selection changes from reverse (*i.e.,* if PRNDL != 1) at any time in the ENGAGED superstate, RA becomes DISENGAGED. A driver acceleration pedal input exceeding 35% of depression at any time will cause RA to enter the OVERRIDE state until the input becomes below the threshold, which causes RA to go back to the ENABLED state as long as the feature is still enabled. An Error event at any time when RA is on will send the feature to the FAILED state, where RA will remain until the car is turned off.

The definition of stable for UWFMS's RA is defined as 1, *i.e., true* because the only one transition is taken in each big-step as ordered-compositions are not present in RA.

Type	**Name**	**Meaning**
event	Error	A signal indicating when an error has occurred
data	RA_Enabled	Driver selected main power to enable/disable the feature [Boolean]
data	BrakePedal	Value indicating the physical amount of depression of the brake pedal by the driver (0 for not depressed and any positive value when depressed) [Percentage in integers]
data	AccelPedal	Value of physical pedal input represented as a percentage of maximum depression [Percentage in integers]
data	Speed	Current speed of the vehicle (value within the range of 0 to 100) [KPH in integers]

data	ObstacleZone	Input from a pre-processing threat assessment block that converts sensor inputs into a threat:
		Threat = 0: No Obstacle
		Threat = 1: Mild Threat
		Threat = 2: Imminent Collision Threat
		[Threat value in integers]

data	PRNDL_In	The input representing the current gear selection with the following values assumed:
		PRNDL = 0: Park
		PRNDL = 1: Reverse
		PRNDL = 2: Neutral
		PRNDL = 3: Drive
		PRNDL = 4: Low
		[Gear selection in integers]

Table A.14: Input variables used in Reversing Assistant (RA)

Type	**Name**	**Meaning**
data	set_Brake	An output request for braking force by the feature as a percentage of maximum braking ability (For RA, only one value of braking is output, which corresponds to mid-force braking) [Percentage in integers]

data	RA_HVI	Value to indicate the message being displayed to the driver through a human-vehicle-interface (HVI):
		RA_HVI = 0: CA Disabled
		RA_HVI = 1: CA Enabled
		RA_HVI = 2: CA Engaged
		RA_HVI = 3: CA Error
		RA_HVI = 4: CA Override
		[Display value in integers]

data	RA_Warning	Audible and/or visual warning to indicate the presence of threats, their severity and the intervention of RA:
		Warning = 0: No Obstacle
		Warning = 1: Mild Threat
		Warning = 2: Imminent Collision Threat
		Warning = 3: Vehicle Held Stopped
		[Warning value in integers]

Table A.15: Output variables used in Reversing Assistant (RA)

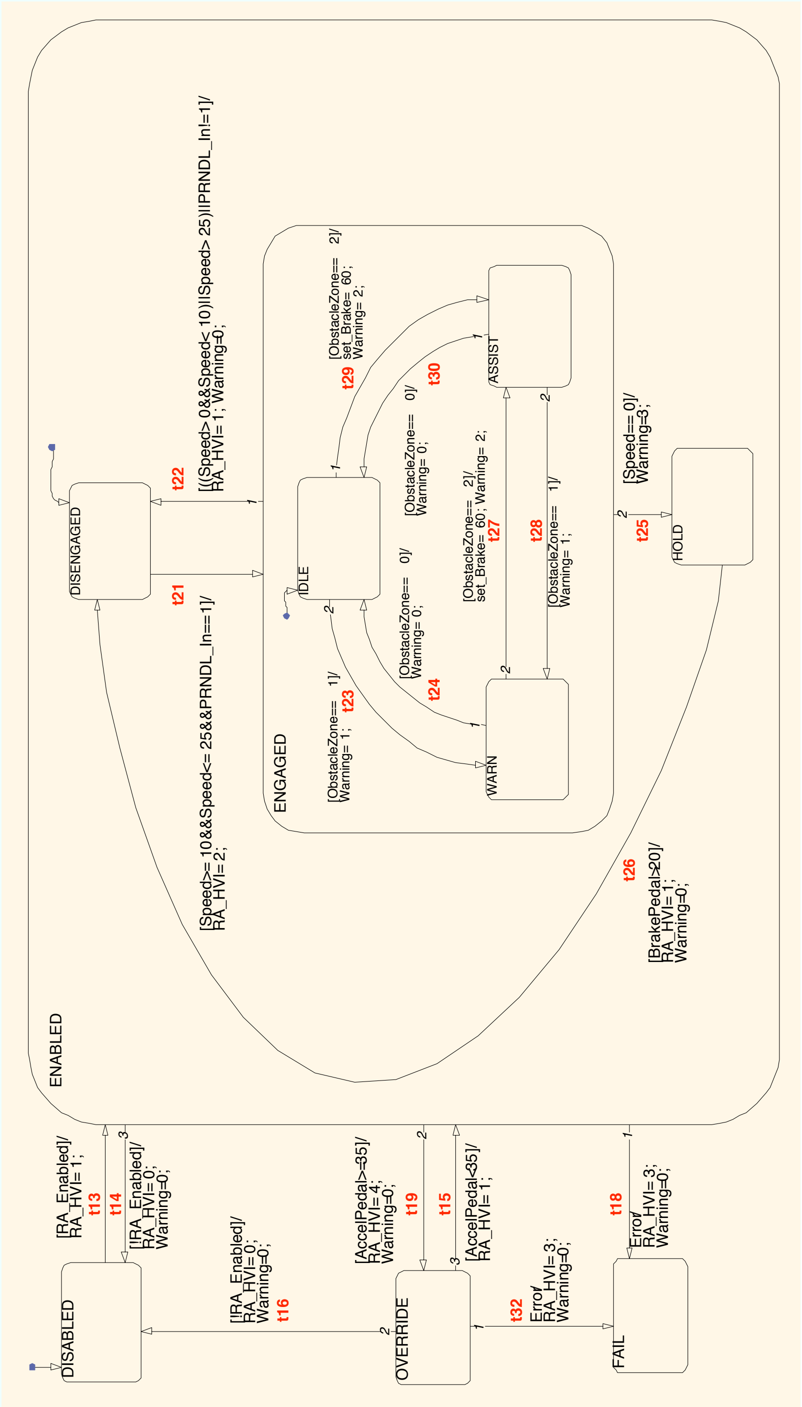


Figure A.11: Reversing Assistant (RA) Stateflow design model

A.8 Summary

My research effort is helped by producing the base for future feature interaction analysis, creating a set of non-proprietary automotive feature design models in Stateflow, called "University of Waterloo Feature Model Set" (UWFMS). This feature model set was created because there was no other set that I could have used to validate my methods and tools. The feature design models part of the UWFMS follow the syntactic modelling rules described in Section 4.2. Many of the design decision made were influenced by my interaction with the design engineers during my visits to GM Research and Development.