# PVS by Example[1]

César A. Muñoz

NASA's Langley Research Center
`Cesar.A.Munoz@nasa.gov`

Formal Proofs in PVS – Tutorial
18th Conference on Intelligent Computer Mathematics
October 6, 2025

[1] Approved for public release STRIVES # 20250009059

# The Prototype Verification System (PVS)

PVS consists of a specification language to write formal models of

- system requirements (internal and external),
- functional algorithms

and a (highly automated) interactive theorem prover to verify that these models are *correct*.

To have in mind:

> *All models are wrong; the practical question is how wrong they have to be to not be useful*[2]

[2] G. Box and N. Draper, Empirical Model Building and Response Surfaces, 1987.

# The Prototype Verification System (PVS)

PVS consists of a specification language to write formal models of

- system requirements (internal and external),
- functional algorithms

and a (highly automated) interactive theorem prover to verify that these models are *correct*.

To have in mind:

> *All models are wrong; the practical question is how wrong they have to be to not be useful*[2]

[2] G. Box and N. Draper, Empirical Model Building and Response Surfaces, 1987.

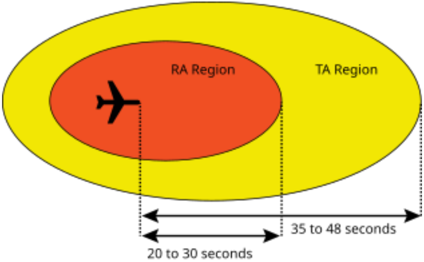
# PVS in a Nutshell

- PVS[3] is developed by SRI International with the support of the Formal Methods Team at NASA's Langley Research Center
- Strongly typed specification language based on classical higher-order logic
- Extensible theorem prover with built-in decision procedures and soundness-preserving strategy language
- Modern graphical development interface[4] based on Microsoft Visual Studio Code
- De-facto library[5] consisting of 53 libraries and about 30K formally proven lemmas
- Batch proving and animation capabilities

[3] `http://pvs.csl.sri.com`
[4] `https://shemesh.larc.nasa.gov/fm/VSCode-PVS`
[5] `https://shemesh.larc.nasa.gov/fm/pvs/PVS-library`

# Traffic Alert and Collision Avoidance System (TCAS)

- Family of of airborne systems designed to reduce the risk of mid-air collisions between *cooperative* aircraft.
- Mandated in the US for aircraft with greater than 30 seats or a maximum takeoff weight greater than 33,000 pounds.
- Current version, TCAS II V7.1, provides[6]:
  - **Traffic Alerts** (TAs).
  - **(Vertical) Resolution Advisories** (RAs).




[6]Graphics released under Creative Commons License (`https://en.wikipedia.org/wiki/Traffic_collision_avoidance_system`)

# TCAS Alerting Logic

- TCAS alerting logic is pairwise, i.e., it provides alerting information for one aircraft (the ownship) against a traffic aircraft (the intruder).
- TCAS logic uses 3-dimensional tracking of aircraft position and velocity.
- TCAS logic predicts aircraft trajectories using a linear projection of current aircraft position and velocity.
- Predicted trajectories are checked against time and distance thresholds, whose values depend on sensitivity level.

# TCAS 2D Core Alerting Logic

(Simplified 2D version of the logic!)

A Traffic Alert (TA) is issued in the ownship aircraft, with respect to an intruder aircraft, when

- The range between the two aircraft is below DMOD threshold for ownship sensitivity level **or**
- Aircraft are converging **and** time Tau between the two aircraft is below TAU threshold for ownship sensitivity level.[7]

Henceforth, it will be assumed that both aircraft have the same sensitivity level.

[7] Tau is related to the time to closest point approach (TCPA).

# Range, Closure Rate, and $\tau$

- Range ($r$): Horizontal distance between 2 aircraft.
- Closure rate ($-\dot{r}$): Negative of range rate.
- Tau ($\tau$): Range over closure rate

$$\tau \equiv -\frac{r}{\dot{r}}$$

# Aircraft State Information

## 2D Euclidean Airspace

- $\mathbf{s}_o, \mathbf{v}_o$: Ownship's current position and velocity (2D Vectors).
- $\mathbf{s}_i, \mathbf{v}_i$: Intruder's current position and velocity (2D Vectors).
- It is convenient to express aircraft states in a relative coordinate system where the intruder is fixed at the center:
  - $\mathbf{s} = \mathbf{s}_o - \mathbf{s}_i$
  - $\mathbf{v} = \mathbf{v}_o - \mathbf{v}_i$

$$
\begin{aligned}
r(\mathbf{s}) &\equiv \|\mathbf{s}\|, \\
\dot{r}(\mathbf{s}, \mathbf{v}) &\equiv \frac{\mathbf{s} \cdot \mathbf{v}}{\|\mathbf{s}\|} \\
\texttt{converging?}(\mathbf{s}, \mathbf{v}) &\equiv \mathbf{s} \cdot \mathbf{v} < 0 \\
\tau(\mathbf{s}, \mathbf{v}) &\equiv -\frac{\mathbf{s}^2}{\mathbf{s} \ \cdot \ \mathbf{v}}
\end{aligned}
$$

# ...in PVS

```
TCAS_tau : THEORY
BEGIN

  IMPORTING vectors@vectors_2D

  % s is a 2D relative position
  % v is a 2D relative velocity
  s,v : VAR Vect2

  range(s) : nnreal = norm(s)

  closure_rate(nzs:Nz_vect2,v): real =
    -(nzs*v)/norm(nzs)

  ...
END TCAS_tau
```

# . . . in PVS

```
TCAS_tau : THEORY
BEGIN

  IMPORTING vectors@vectors_2D

  % s is a 2D relative position
  % v is a 2D relative velocity
  s,v : VAR Vect2

  range(s) : nnreal = norm(s)

  closure_rate(nzs:Nz_vect2,v): real =
    -(nzs*v)/norm(nzs)

  ...
END TCAS_tau
```

## ...in PVS

```
TCAS_tau : THEORY
BEGIN

  IMPORTING vectors@vectors_2D

  % s is a 2D relative position
  % v is a 2D relative velocity
  s,v : VAR Vect2

  range(s) : nnreal = norm(s)

  closure_rate(nzs:Nz_vect2,v): real =
    -(nzs*v)/norm(nzs)

  ...
END TCAS_tau
```

# Converging Aircraft

```
converging?(s)(v) :bool =
  s*v < 0

% nzv is a non-zero 2D vector
nzs : VAR Nz_vect2

converging_closure_rate : LEMMA
  closure_rate(nzs,v) > 0 IFF converging?(nzs)(v)
```

```
%|- converging_closure_rate : PROOF
%|- (then (skeep)
%|-   (spread (grind) ((grind-reals) (grind-reals))))
%|- QED converging_closure_rate
```

# Converging Aircraft

```
converging?(s)(v) :bool =
  s*v < 0

% nzv is a non-zero 2D vector
nzs : VAR Nz_vect2

converging_closure_rate : LEMMA
  closure_rate(nzs,v) > 0 IFF converging?(nzs)(v)
```

```
%|- converging_closure_rate : PROOF
%|- (then (skeep)
%|-   (spread (grind) ((grind-reals) (grind-reals))))
%|- QED converging_closure_rate
```

# Converging Aircraft

```
converging?(s)(v) :bool =
  s*v < 0

% nzv is a non-zero 2D vector
nzs : VAR Nz_vect2

converging_closure_rate : LEMMA
  closure_rate(nzs,v) > 0 IFF converging?(nzs)(v)
```

```
%|- converging_closure_rate : PROOF
%|- (then (skeep)
%|-   (spread (grind) ((grind-reals) (grind-reals))))
%|- QED converging_closure_rate
```

# Converging Aircraft

```
converging?(s)(v) :bool =
  s*v < 0

% nzv is a non-zero 2D vector
nzs : VAR Nz_vect2

converging_closure_rate : LEMMA
  closure_rate(nzs,v) > 0 IFF converging?(nzs)(v)
```

```
%|- converging_closure_rate : PROOF
%|- (then (skeep)
%|-   (spread (grind) ((grind-reals) (grind-reals))))
%|- QED converging_closure_rate
```

# TCAS Tau

```
% Time tau is only defined when aircraft are converging
tau(s:Vect2,v:(converging?(s))) : nnreal =
  -sqv(s)/(s*v)
```

- Type Correctness Conditions:

```
tau_TCC1: OBLIGATION
  FORALL (s: Vect2, v: (converging?(s))):
    (s*v) /= 0;


tau_TCC2: OBLIGATION
  FORALL (s: Vect2, v: (converging?(s))):
    -sqv(s)/(s*v) >= 0;
```

- TCAS Tau is range over closure rate

```
tau_def : LEMMA
  converging?(s)(v) IMPLIES
  tau(s,v) = range(s)/closure_rate(s,v)
```

# TCAS Tau

```
% Time tau is only defined when aircraft are converging
tau(s:Vect2,v:(converging?(s))) : nnreal =
  -sqv(s)/(s*v)
```

- Type Correctness Conditions:

```
tau_TCC1: OBLIGATION
  FORALL (s: Vect2, v: (converging?(s))):
    (s*v) /= 0;
```

```
tau_TCC2: OBLIGATION
  FORALL (s: Vect2, v: (converging?(s))):
    -sqv(s)/(s*v) >= 0;
```

- TCAS Tau is range over closure rate

```
tau_def : LEMMA
  converging?(s)(v) IMPLIES
  tau(s,v) = range(s)/closure_rate(s,v)
```

# TCAS Tau

```
% Time tau is only defined when aircraft are converging
tau(s:Vect2,v:(converging?(s))) : nnreal =
  -sqv(s)/(s*v)
```

- Type Correctness Conditions:

```
tau_TCC1: OBLIGATION
  FORALL (s: Vect2, v: (converging?(s))):
    (s*v) /= 0;

tau_TCC2: OBLIGATION
  FORALL (s: Vect2, v: (converging?(s))):
    -sqv(s)/(s*v) >= 0;
```

- TCAS Tau is range over closure rate

```
tau_def : LEMMA
  converging?(s)(v) IMPLIES
  tau(s,v) = range(s)/closure_rate(s,v)
```

# TCAS Tau

```
% Time tau is only defined when aircraft are converging
tau(s:Vect2,v:(converging?(s))) : nnreal =
  -sqv(s)/(s*v)
```

- Type Correctness Conditions:

```
tau_TCC1: OBLIGATION
  FORALL (s: Vect2, v: (converging?(s))):
    (s*v) /= 0;

tau_TCC2: OBLIGATION
  FORALL (s: Vect2, v: (converging?(s))):
    -sqv(s)/(s*v) >= 0;
```

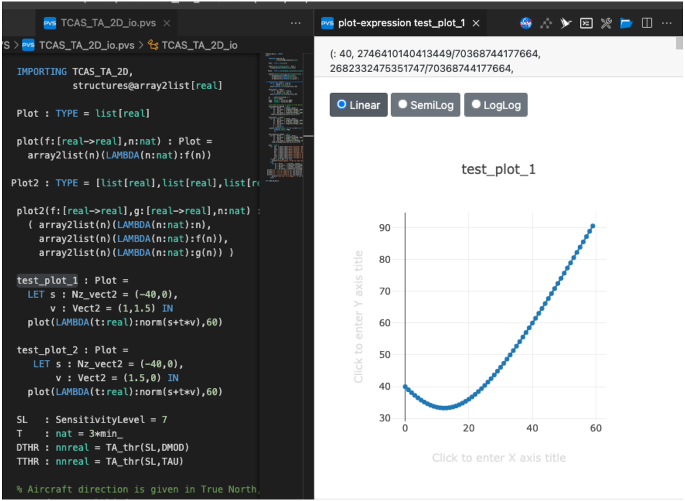
- TCAS Tau is range over closure rate

```
tau_def : LEMMA
   converging?(s)(v) IMPLIES
   tau(s,v) = range(s)/closure_rate(s,v)
```

# Range as a Function of Time

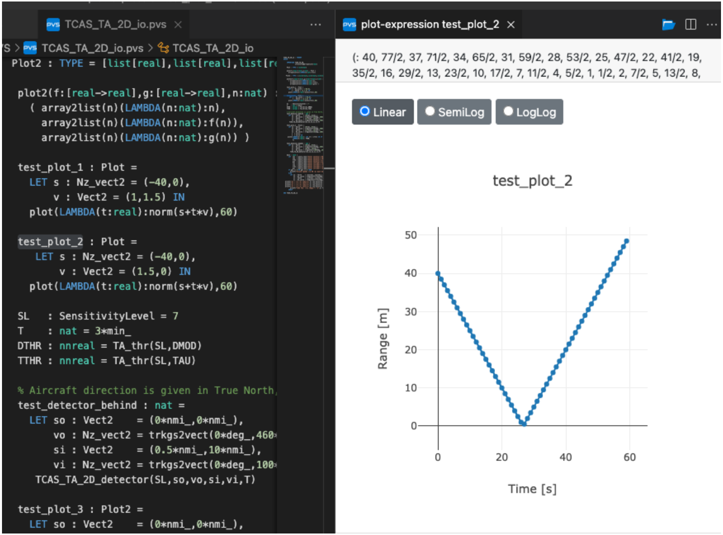
Let **s** and **v** be initial position and constant velocity:

$t \mapsto \|\mathbf{s} + t * \mathbf{v}\|$.




(Used with permission from Microsoft.)

# Derivative of Range May Not Exist!




(Used with permission from Microsoft.)

# Subtyping, Dependent Types, Auto Rewrites, Judgements

```
RangeDomain(nzs,v) : TYPE =
  {t:real| det(nzs,v) = 0 IMPLIES t*sqv(v) /= -nzs*v}

range_deriv_domain : LEMMA
  deriv_domain?[RangeDomain(nzs,v)]

AUTO_REWRITE+ range_deriv_domain

range_at(nzs,v)(t:RangeDomain(nzs, v)) : nnreal =
  range(nzs+t*v)

derivable_range_at : JUDGEMENT
  range_at(nzs,v) HAS_TYPE
    (differentiable?[RangeDomain(nzs, v)])
```

# Subtyping, Dependent Types, Auto Rewrites, Judgements

```
RangeDomain(nzs,v) : TYPE =
  {t:real| det(nzs,v) = 0 IMPLIES t*sqv(v) /= -nzs*v}

range_deriv_domain : LEMMA
  deriv_domain?[RangeDomain(nzs,v)]

AUTO_REWRITE+ range_deriv_domain

range_at(nzs,v)(t:RangeDomain(nzs, v)) : nnreal =
  range(nzs+t*v)

derivable_range_at : JUDGEMENT
  range_at(nzs,v) HAS_TYPE
    (differentiable?[RangeDomain(nzs, v)])
```

# Subtyping, Dependent Types, Auto Rewrites, Judgements

```
RangeDomain(nzs,v) : TYPE =
  {t:real| det(nzs,v) = 0 IMPLIES t*sqv(v) /= -nzs*v}

range_deriv_domain : LEMMA
  deriv_domain?[RangeDomain(nzs,v)]

AUTO_REWRITE+ range_deriv_domain

range_at(nzs,v)(t:RangeDomain(nzs, v)) : nnreal =
  range(nzs+t*v)

derivable_range_at : JUDGEMENT
  range_at(nzs,v) HAS_TYPE
    (differentiable?[RangeDomain(nzs, v)])
```

# Subtyping, Dependent Types, Auto Rewrites, Judgements

```
RangeDomain(nzs,v) : TYPE =
  {t:real| det(nzs,v) = 0 IMPLIES t*sqv(v) /= -nzs*v}

range_deriv_domain : LEMMA
  deriv_domain?[RangeDomain(nzs,v)]

AUTO_REWRITE+ range_deriv_domain

range_at(nzs,v)(t:RangeDomain(nzs, v)) : nnreal =
  range(nzs+t*v)

derivable_range_at : JUDGEMENT
  range_at(nzs,v) HAS_TYPE
    (differentiable?[RangeDomain(nzs, v)])
```

# Subtyping, Dependent Types, Auto Rewrites, Judgements

```
RangeDomain(nzs,v) : TYPE =
  {t:real| det(nzs,v) = 0 IMPLIES t*sqv(v) /= -nzs*v}

range_deriv_domain : LEMMA
  deriv_domain?[RangeDomain(nzs,v)]

AUTO_REWRITE+ range_deriv_domain

range_at(nzs,v)(t:RangeDomain(nzs, v)) : nnreal =
  range(nzs+t*v)

derivable_range_at : JUDGEMENT
  range_at(nzs,v) HAS_TYPE
    (differentiable?[RangeDomain(nzs, v)])
```

# Derivatives

```
%|- derivable_range_at : PROOF
%|- (then (skeep) (rewrite "range_at_eq")
%|=   (derivable :use "sq_range_at")
%|-  (hide 2) (skeep :preds? t) (assert))
%|- QED derivable_range_at

  derivative_range_at : LEMMA
    deriv[RangeDomain(nzs,v)](range_at(nzs,v)) =
    LAMBDA(t:RangeDomain(nzs,v)):(nzs*v+t*sqv(v))/norm(nzs+t*v)

%|- derivative_range_at : PROOF
%|- (then (skeep) (rewrite "range_at_eq")
%|-  (spread (deriv :use "sq_range_at")
%|-   ((then (expand "sq_range_at" 1)(expand "norm" 1)(propax))
%|-    (then (hide -)(skeep)(lemma "range_domain_pos")(inst?)
%|-     (expand "norm") (assert)))))
%|- QED derivative_range_at
```

# Derivatives

```
%|- derivable_range_at : PROOF
%|- (then (skeep) (rewrite "range_at_eq")
%|=   (derivable :use "sq_range_at")
%|-  (hide 2) (skeep :preds? t) (assert))
%|- QED derivable_range_at

  derivative_range_at : LEMMA
    deriv[RangeDomain(nzs,v)](range_at(nzs,v)) =
    LAMBDA(t:RangeDomain(nzs,v)):(nzs*v+t*sqv(v))/norm(nzs+t*v)

%|- derivative_range_at : PROOF
%|- (then (skeep) (rewrite "range_at_eq")
%|-  (spread (deriv :use "sq_range_at")
%|-   ((then (expand "sq_range_at" 1)(expand "norm" 1)(propax))
%|-    (then (hide -)(skeep)(lemma "range_domain_pos")(inst?)
%|-     (expand "norm") (assert)))))
%|- QED derivative_range_at
```

# Closure Rate is the Neg. Derivative of Range Over Time

```
closure_rate_def : LEMMA
  (det(nzs,v) /= 0 OR v /= zero) IMPLIES
  closure_rate(nzs,v) =
  -deriv[RangeDomain(nzs,v)](range_at(nzs,v))(0)

%|- closure_rate_def : PROOF
%|- (then (skeep)(rewrite "derivative_range_at")(beta)
%|-   (expand "closure_rate")(assert))
%|-  QED closure_rate_def
```

# TCAS TA 2D Core Logic

```
TCAS_TA_2D_core[DTHR,TTHR:nnreal]: THEORY
BEGIN

  IMPORTING TCAS_tau

  TCAS_TA_2D_core(s,v:Vect2) : bool =
   LET s2 = sqv(s) IN
     s2  < sq(DTHR) OR
     (converging?(s)(v) AND s2 < -TTHR*(s*v))

  % Requirement: Both aircraft have the same TA status
  TCAS_TA_2D_core_sym : LEMMA
    FORALL (s,v:Vect2):
      TCAS_TA_2D_core(s,v) = TCAS_TA_2D_core(-s,-v)
...
END TCAS_TA_2D_core
```

# TCAS TA 2D Core Logic

```
TCAS_TA_2D_core[DTHR,TTHR:nnreal]: THEORY
BEGIN

  IMPORTING TCAS_tau

  TCAS_TA_2D_core(s,v:Vect2) : bool =
   LET s2 = sqv(s) IN
     s2  < sq(DTHR) OR
     (converging?(s)(v) AND s2 < -TTHR*(s*v))

  % Requirement: Both aircraft have the same TA status
  TCAS_TA_2D_core_sym : LEMMA
    FORALL (s,v:Vect2):
      TCAS_TA_2D_core(s,v) = TCAS_TA_2D_core(-s,-v)
...
END TCAS_TA_2D_core
```

# TCAS TA 2D Core Logic

```
TCAS_TA_2D_core[DTHR,TTHR:nnreal]: THEORY
BEGIN

  IMPORTING TCAS_tau

  TCAS_TA_2D_core(s,v:Vect2) : {b : bool | b IFF
          norm(s) < DTHR OR
          (converging?(s)(v) AND tau(s,v) < TTHR)} =
   LET s2 = sqv(s) IN
     s2  < sq(DTHR) OR
     (converging?(s)(v) AND s2 < -TTHR*(s*v))

  % Requirement: Both aircraft have the same TA status
  TCAS_TA_2D_core_sym : LEMMA
    FORALL (s,v:Vect2):
      TCAS_TA_2D_core(s,v) = TCAS_TA_2D_core(-s,-v)
...
END TCAS_TA_2D_core
```

# Types as Specifications

```
% Returns first time when a TA is issued up to T
% or T+1 if TA won't be issued
TCAS_TA_2D_det_rec(s,v:Vect2,T:nat)(t:upto(T+1)) :
  RECURSIVE upto(T+1) =
  IF t > T OR TCAS_TA_2D_core(s+t*v,v) THEN t
  ELSE TCAS_TA_2D_det_rec(s,v,T)(t+1)
  ENDIF
MEASURE T+1-t

TCAS_TA_2D_detector(s,v:Vect2,T:nat): {k:upto(T+1) |
  (FORALL (i:below(k)) : NOT TCAS_TA_2D_core(s+i*v,v)) AND
                 (k <= T IMPLIES  TCAS_TA_2D_core(s+k*v,v))} =
  TCAS_TA_2D_det_rec(s,v,T)(0)
```

# Types as Specifications

```
% Returns first time when a TA is issued up to T
% or T+1 if TA won't be issued
TCAS_TA_2D_det_rec(s,v:Vect2,T:nat)(t:upto(T+1) |
    FORALL (i:below(t)) : NOT TCAS_TA_2D_core(s+i*v,v)) :
  RECURSIVE upto(T+1) =
  IF t > T OR TCAS_TA_2D_core(s+t*v,v) THEN t
  ELSE TCAS_TA_2D_det_rec(s,v,T)(t+1)
  ENDIF
MEASURE T+1-t

TCAS_TA_2D_detector(s,v:Vect2,T:nat): {k:upto(T+1) |
  (FORALL (i:below(k)) : NOT TCAS_TA_2D_core(s+i*v,v)) AND
               (k <= T IMPLIES  TCAS_TA_2D_core(s+k*v,v))} =
  TCAS_TA_2D_det_rec(s,v,T)(0)
```

# Types as Specifications

```
% Returns first time when a TA is issued up to T
% or T+1 if TA won't be issued
TCAS_TA_2D_det_rec(s,v:Vect2,T:nat)(t:upto(T+1) |
    FORALL (i:below(t)) : NOT TCAS_TA_2D_core(s+i*v,v)) :
  RECURSIVE {k:upto(T+1) |
    (FORALL (i:below(k)) : NOT TCAS_TA_2D_core(s+i*v,v)) AND
               (k <= T IMPLIES  TCAS_TA_2D_core(s+k*v,v))} =
  IF t > T OR TCAS_TA_2D_core(s+t*v,v) THEN t
  ELSE TCAS_TA_2D_det_rec(s,v,T)(t+1)
  ENDIF
MEASURE T+1-t

TCAS_TA_2D_detector(s,v:Vect2,T:nat): {k:upto(T+1) |
  (FORALL (i:below(k)) : NOT TCAS_TA_2D_core(s+i*v,v)) AND
               (k <= T IMPLIES  TCAS_TA_2D_core(s+k*v,v))} =
  TCAS_TA_2D_det_rec(s,v,T)(0)
```

# Sensitivity Levels and Thresholds

| **Ownship Altitude** (feet) | **SL** | **TAU** (sec) | **DMOD** (nmi) | **ZTHR** (feet) |
|---|---|---|---|---|
| **Below 1000** | 2 | 20 | 0.30 | 850 |
| **1000 - 2350** | 3 | 25 | 0.33 | 850 |
| **2350 - 5000** | 4 | 30 | 0.48 | 850 |
| **5000 -10000** | 5 | 40 | 0.75 | 850 |
| **10000 - 20000** | 6 | 45 | 1.0 | 850 |
| **20000 - 42000** | 7 | 48 | 1.3 | 850 |
| **Above 42000** | 8 | 48 | 1.3 | 1200 |

# Sensitivity Levels and Thresholds

| **Ownship Altitude** (feet) | **SL** | **TAU** (sec) | **DMOD** (nmi) | **ZTHR** (feet) |
|---|---|---|---|---|
| **Below 1000** | 2 | 20 | 0.30 | 850 |
| **1000 - 2350** | 3 | 25 | 0.33 | 850 |
| **2350 - 5000** | 4 | 30 | 0.48 | 850 |
| **5000 -10000** | 5 | 40 | 0.75 | 850 |
| **10000 - 20000** | 6 | 45 | 1.0 | 850 |
| **20000 - 42000** | 7 | 48 | 1.3 | 850 |
| **Above 42000** | 8 | 48 | 1.3 | 1200 |

# A Basic Theory of Units

```
Units : THEORY
BEGIN
  m_    :  posreal = 1          % 1 meter
  km_   :  posreal = 1000       % 1 kilometer in meters
  ft_   :  posreal = 0.3048     % 1 foot in meters
  nmi_  :  posreal = 1852       % 1 nautical mile in meters

  s_    :  posreal = 1          % 1 second
  min_  :  posreal = 60         % 1 minute in seconds
  hour_ :  posreal = 60*min_    % 1 hour in seconds

  knt_  :  posreal = nmi_/hour_ % 1 knot in m/s
  fpm_  :  posreal = ft_/min_   % 1 foot per minute in m/s
  kph_  :  posreal = km_/hour_  % 1 km per hour in m/s
  ...
  to_units(val:real,unit:posreal) : real = val/unit
  ...
END Units
```

# A Basic Theory of Units

```
Units : THEORY
BEGIN
  m_    :  posreal = 1          % 1 meter
  km_   :  posreal = 1000       % 1 kilometer in meters
  ft_   :  posreal = 0.3048     % 1 foot in meters
  nmi_  :  posreal = 1852       % 1 nautical mile in meters

  s_    :  posreal = 1          % 1 second
  min_  :  posreal = 60         % 1 minute in seconds
  hour_ :  posreal = 60*min_    % 1 hour in seconds

  knt_  :  posreal = nmi_/hour_ % 1 knot in m/s
  fpm_  :  posreal = ft_/min_   % 1 foot per minute in m/s
  kph_  :  posreal = km_/hour_  % 1 km per hour in m/s
  ...
  to_units(val:real,unit:posreal) : real = val/unit
  ...
END Units
```

# A Basic Theory of Units

```
Units : THEORY
BEGIN
  m_    : MACRO posreal = 1         % 1 meter
  km_   : MACRO posreal = 1000      % 1 kilometer in meters
  ft_   : MACRO posreal = 0.3048    % 1 foot in meters
  nmi_  : MACRO posreal = 1852      % 1 nautical mile in meters

  s_    : MACRO posreal = 1         % 1 second
  min_  : MACRO posreal = 60        % 1 minute in seconds
  hour_ : MACRO posreal = 60*min_   % 1 hour in seconds

  knt_  : MACRO posreal = nmi_/hour_ % 1 knot in m/s
  fpm_  : MACRO posreal = ft_/min_   % 1 foot per minute in m/s
  kph_  : MACRO posreal = km_/hour_  % 1 km per hour in m/s
  ...
  to_units(val:real,unit:posreal) : real = val/unit
  ...
END Units
```

# A Basic Theory of Units

```
Units : THEORY
BEGIN
  m_    : MACRO posreal = 1          % 1 meter
  km_   : MACRO posreal = 1000       % 1 kilometer in meters
  ft_   : MACRO posreal = 0.3048     % 1 foot in meters
  nmi_  : MACRO posreal = 1852       % 1 nautical mile in meters

  s_    : MACRO posreal = 1          % 1 second
  min_  : MACRO posreal = 60         % 1 minute in seconds
  hour_ : MACRO posreal = 60*min_    % 1 hour in seconds

  knt_  : MACRO posreal = nmi_/hour_ % 1 knot in m/s
  fpm_  : MACRO posreal = ft_/min_   % 1 foot per minute in m/s
  kph_  : MACRO posreal = km_/hour_  % 1 km per hour in m/s
  ...
  to_units(val:real,unit:posreal) : real = val/unit
  ...
END Units
```

# TCAS Tables: Sensitivity Level

```
TCAS_tables : THEORY
BEGIN
  IMPORTING Units

  SensitivityLevel : TYPE = subrange(2,8)

  sensitivity_level(alt:nnreal) : SensitivityLevel =
    TABLE
    %+--------------------------------------+---++
     |     0*ft_ <= alt AND alt <  1000*ft_ | 2 ||
    %+--------------------------------------+---++
     |  1000*ft_ <= alt AND alt <  2350*ft_ | 3 ||
    %+--------------------------------------+---++
     |  2350*ft_ <= alt AND alt <  5000*ft_ | 4 ||
    %+--------------------------------------+---++
     |  5000*ft_ <= alt AND alt < 10000*ft_ | 5 ||
    %+--------------------------------------+---++
     | 10000*ft_ <= alt AND alt < 20000*ft_ | 6 ||
    %+--------------------------------------+---++
     | 20000*ft_ <= alt AND alt < 42000*ft_ | 7 ||
    %+--------------------------------------+---++
     | ELSE                                 | 8 ||
    %+--------------------------------------+---++
    ENDTABLE
```

# TCAS Tables: Sensitivity Level

```
TCAS_tables : THEORY
BEGIN
  IMPORTING Units

  SensitivityLevel : TYPE = subrange(2,8)

  sensitivity_level(alt:nnreal) : SensitivityLevel =
    TABLE
    %+--------------------------------------+---++
     |     0*ft_ <= alt AND alt <  1000*ft_ | 2 ||
    %+--------------------------------------+---++
     |  1000*ft_ <= alt AND alt <  2350*ft_ | 3 ||
    %+--------------------------------------+---++
     |  2350*ft_ <= alt AND alt <  5000*ft_ | 4 ||
    %+--------------------------------------+---++
     |  5000*ft_ <= alt AND alt < 10000*ft_ | 5 ||
    %+--------------------------------------+---++
     | 10000*ft_ <= alt AND alt < 20000*ft_ | 6 ||
    %+--------------------------------------+---++
     | 20000*ft_ <= alt AND alt < 42000*ft_ | 7 ||
    %+--------------------------------------+---++
     | ELSE                                 | 8 ||
    %+--------------------------------------+---++
    ENDTABLE
```

# TCAS Tables: Sensitivity Level

```
TCAS_tables : THEORY
BEGIN
  IMPORTING Units

  SensitivityLevel : TYPE = subrange(2,8)

  sensitivity_level(alt:nnreal) : SensitivityLevel =
    TABLE
    %+-----------------------------------------+---++
     |     0*ft_ <= alt AND alt <  1000*ft_ | 2 ||
    %+-----------------------------------------+---++
     |  1000*ft_ <= alt AND alt <  2350*ft_ | 3 ||
    %+-----------------------------------------+---++
     |  2350*ft_ <= alt AND alt <  5000*ft_ | 4 ||
    %+-----------------------------------------+---++
     |  5000*ft_ <= alt AND alt < 10000*ft_ | 5 ||
    %+-----------------------------------------+---++
     | 10000*ft_ <= alt AND alt < 20000*ft_ | 6 ||
    %+-----------------------------------------+---++
     | 20000*ft_ <= alt AND alt < 42000*ft_ | 7 ||
    %+-----------------------------------------+---++
     | ELSE                                 | 8 ||
    %+-----------------------------------------+---++
    ENDTABLE
```

# TCAS Tables: Sensitivity Level

```
TCAS_tables : THEORY
BEGIN
  IMPORTING Units

  SensitivityLevel : TYPE = subrange(2,8)

  sensitivity_level(alt:nnreal) : SensitivityLevel =
    TABLE
    %+---------------------------------------+---++
     |     0*ft_ <= alt AND alt <  1000*ft_ | 2 ||
    %+---------------------------------------+---++
     |  1000*ft_ <= alt AND alt <  2350*ft_ | 3 ||
    %+---------------------------------------+---++
     |  2350*ft_ <= alt AND alt <  5000*ft_ | 4 ||
    %+---------------------------------------+---++
     |  5000*ft_ <= alt AND alt < 10000*ft_ | 5 ||
    %+---------------------------------------+---++
     | 10000*ft_ <= alt AND alt < 20000*ft_ | 6 ||
    %+---------------------------------------+---++
     | 20000*ft_ <= alt AND alt < 42000*ft_ | 7 ||
    %+---------------------------------------+---++
     | ELSE                                  | 8 ||
    %+---------------------------------------+---++
    ENDTABLE
```

# TCAS Tables: Sensitivity Level

```
TCAS_tables : THEORY
BEGIN
  IMPORTING Units

  SensitivityLevel : TYPE = subrange(2,8)

  sensitivity_level(alt:nnreal) : SensitivityLevel =
    TABLE
    %+--------------------------------------+---++
     |      0*ft_ <= alt AND alt <  1000*ft_ | 2 ||
    %+--------------------------------------+---++
     |   1000*ft_ <= alt AND alt <  2350*ft_ | 3 ||
    %+--------------------------------------+---++
     |   2350*ft_ <= alt AND alt <  5000*ft_ | 4 ||
    %+--------------------------------------+---++
     |   5000*ft_ <= alt AND alt < 10000*ft_ | 5 ||
    %+--------------------------------------+---++
     |  10000*ft_ <= alt AND alt < 20000*ft_ | 6 ||
    %+--------------------------------------+---++
     |  20000*ft_ <= alt AND alt < 42000*ft_ | 7 ||
    %+--------------------------------------+---++
     | ELSE                                  | 8 ||
    %+--------------------------------------+---++
    ENDTABLE
```

# TCAS Tables: TAU, DMOD, and ZTHR

```
ThresholdSymbol : TYPE = { TAU, DMOD, ZTHR }

TA_thr(sl:SensitivityLevel,thr:ThresholdSymbol) : nnreal =
   TABLE sl ,      thr
      %--- +------+-----------+----------++
           |[ TAU |    DMOD   |   ZTHR   ]|
      %--- +------+-----------+----------++
       | 2 |   20 | 0.30*nmi_ |  850*ft_ ||
      %--- +------+-----------+----------++
       | 3 |   25 | 0.33*nmi_ |  850*ft_ ||
      %--- +------+-----------+----------++
       | 4 |   30 | 0.48*nmi_ |  850*ft_ ||
      %--- +------+-----------+----------++
       | 5 |   40 | 0.75*nmi_ |  850*ft_ ||
      %--- +------+-----------+----------++
       | 6 |   45 | 1.0*nmi_  |  850*ft_ ||
      %--- +------+-----------+----------++
       | 7 |   48 | 1.3*nmi_  |  850*ft_ ||
      %--- +------+-----------+----------++
       | 8 |   48 | 1.3*nmi_  | 1200*ft_ ||
      %--- +------+-----------+----------++
   ENDTABLE
```

# TCAS Tables: TAU, DMOD, and ZTHR

```
ThresholdSymbol : TYPE = { TAU, DMOD, ZTHR }

TA_thr(sl:SensitivityLevel,thr:ThresholdSymbol) : nnreal =
   TABLE sl ,      thr
       %--- +------+-----------+-----------++
            |[ TAU |    DMOD   |    ZTHR   ]|
       %--- +------+-----------+-----------++
        | 2 |   20 | 0.30*nmi_ |  850*ft_ ||
       %--- +------+-----------+-----------++
        | 3 |   25 | 0.33*nmi_ |  850*ft_ ||
       %--- +------+-----------+-----------++
        | 4 |   30 | 0.48*nmi_ |  850*ft_ ||
       %--- +------+-----------+-----------++
        | 5 |   40 | 0.75*nmi_ |  850*ft_ ||
       %--- +------+-----------+-----------++
        | 6 |   45 | 1.0*nmi_  |  850*ft_ ||
       %--- +------+-----------+-----------++
        | 7 |   48 | 1.3*nmi_  |  850*ft_ ||
       %--- +------+-----------+-----------++
        | 8 |   48 | 1.3*nmi_  | 1200*ft_ ||
       %--- +------+-----------+-----------++
   ENDTABLE
```

# TCAS Tables: TAU, DMOD, and ZTHR

```
ThresholdSymbol : TYPE = { TAU, DMOD, ZTHR }

TA_thr(sl:SensitivityLevel,thr:ThresholdSymbol) : nnreal =
   TABLE sl ,     thr
       %--- +------+-----------+----------++
            |[ TAU |    DMOD   |   ZTHR   ]|
       %--- +------+-----------+----------++
        | 2 |   20 | 0.30*nmi_ |  850*ft_ ||
       %--- +------+-----------+----------++
        | 3 |   25 | 0.33*nmi_ |  850*ft_ ||
       %--- +------+-----------+----------++
        | 4 |   30 | 0.48*nmi_ |  850*ft_ ||
       %--- +------+-----------+----------++
        | 5 |   40 | 0.75*nmi_ |  850*ft_ ||
       %--- +------+-----------+----------++
        | 6 |   45 | 1.0*nmi_  |  850*ft_ ||
       %--- +------+-----------+----------++
        | 7 |   48 | 1.3*nmi_  |  850*ft_ ||
       %--- +------+-----------+----------++
        | 8 |   48 | 1.3*nmi_  | 1200*ft_ ||
       %--- +------+-----------+----------++
   ENDTABLE
```

# TCAS Tables: Type Correctness Conditions

```
% Disjointness TCC generated (at line 10, column 4) for
    % TABLE
    %   %------------------------------------------------+---++
    %   | 0 * 0.3048 <= alt AND alt < 1000 * 0.3048      | 2 ||
    ...
    % ENDTABLE
  % proved - complete
sensitivity_level_TCC1: OBLIGATION
  FORALL (alt: nnreal):
       NOT ((0*0.3048 <= alt AND alt < 1000*0.3048) AND
             1000*0.3048 <= alt AND alt < 2350*0.3048) ...

% Coverage TCC generated (at line 34, column 5) for
    % TABLE sl, thr
    %      %+-----+-------------+---------------++
    %      |[ TAU | DMOD        | ZTHR          ]|
    ...
    % ENDTABLE
  % proved - complete
TA_thr_TCC1: OBLIGATION
  FORALL (sl: SensitivityLevel):
      sl=2 OR sl=3 OR sl=4 OR sl=5 OR sl=6 OR sl=7 OR sl=8;
```

# A Maneuver Guidance Algorithm

```
TCAS_TA_2D: THEORY
BEGIN
  IMPORTING TCAS_TA_2D_core, TCAS_tables, aviation@track

  sl     : VAR SensitivityLevel
  so,si  : VAR Vect2    % 2D ownship's and intruder's position
  vo,vi  : VAR Nz_vect2 % 2D ownship's and intruder's velocity
  T      : VAR nat
  gso    : VAR posreal

  TCAS_TA_2D_detector(sl,so,vo,si,vi,T) : upto(T+1) =
    LET DTHR = TA_thr(sl,DMOD),
        TTHR = TA_thr(sl,TAU) IN
    TCAS_TA_2D_detector[DTHR,TTHR](so-si,vo-vi,T)

  % Returns a list of degrees that issues a TA alert
  TCAS_TA_bands_rec(sl,so,gso,si,vi,T)(trk:upto(360)) : RECURSIVE list[below(360)] =
    LET vo = trkgs2vect(trk*deg_,gso) IN
      IF trk = 360 THEN null
      ELSIF TCAS_TA_2D_detector(sl,so,vo,si,vi,T) <= T THEN
        cons(trk,TCAS_TA_bands_rec(sl,so,gso,si,vi,T)(trk+1))
      ELSE TCAS_TA_bands_rec(sl,so,gso,si,vi,T)(trk+1)
      ENDIF
  MEASURE 360-trk

  TCAS_TA_2D_bands(sl,so,vo,si,vi,T) : list[below(360)] =
    TCAS_TA_bands_rec(sl,so,norm(vo),si,vi,T)(0)

END TCAS_TA_2D
```

# Animation of Functional Specifications . . .

```
TCAS_TA_2D_io : THEORY
BEGIN
    SL   : SensitivityLevel = 7
    T    : nat = 3*min_
    DTHR : nnreal = TA_thr(SL,DMOD)
    TTHR : nnreal = TA_thr(SL,TAU)

    % Aircraft direction is given in True North, clockwise convention
    test_detector_behind : nat =
      LET so : Vect2    = (0*nmi_,0*nmi_),
          vo : Nz_vect2 = trkgs2vect(0*deg_,460*knt_),
          si : Vect2    = (0.5*nmi_,10*nmi_),
          vi : Nz_vect2 = trkgs2vect(0*deg_,100*knt_) IN
       TCAS_TA_2D_detector(SL,so,vo,si,vi,T)

    test_detector_headon : nat =
      LET so : Vect2    = (0*nmi_,0*nmi_),
          vo : Nz_vect2 = trkgs2vect(0*deg_,460*knt_),
          si : Vect2    = (0*nmi_,10*nmi_),
          vi : Nz_vect2 = trkgs2vect(180*deg_,100*knt_) IN
        TCAS_TA_2D_detector(SL,so,vo,si,vi,T)
END TCAS_TA_2D_io
```

# . . . in PVSio

```
<PVSio> test_detector_behind;
==>
53

<PVSio> test_detector_headon;
==>
17
```

# Programming in PVSio with I/O

```
main : void =
  LET  sox  = query_real("Enter ownship’s WE position [nmi]:"),
       soy  = query_real("Enter ownship’s SN position [nmi]:"),
       trko = query_real("Enter ownship’s ground track [deg]:"),
       gso  = query_real("Enter ownship’s ground speed [knt]:"),
       six  = query_real("Enter intruder’s WE position [nmi]:"),
       siy  = query_real("Enter intruder’s SN position [nmi]:"),
       trki = query_real("Enter intruder’s ground track [deg]:"),
       gsi  = query_real("Enter intruder’s ground speed [knot]:") IN
  IF gso <= 0 OR gsi <= 0 THEN
    printf("Ground speeds must be strictly positive!")
  ELSE
    LET so : Vect2    = (sox*nmi_,soy*nmi_),
        vo : Nz_vect2 = trkgs2vect(trko*deg_,gso*knt_),
        si : Vect2    = (six*nmi_,siy*nmi_),
        vi : Nz_vect2 = trkgs2vect(trki*deg_,gsi*knt_) IN
   printf("TCAS TAs: ~{~a [deg]~^, ~}",
          {|TCAS_TA_2D_bands(SL,so,vo,si,vi,T)|})
  ENDIF
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# PVSio From Command Line

```
$ pvsio @TCAS_TA_2D_io:
Enter ownship's WE position [nmi]:
0
Enter ownship's SN position [nmi]:
0
Enter ownship's ground track [deg]:
0
Enter ownship's ground speed [knt]:
300
Enter intruder's WE position [nmi]:
10
Enter intruder's SN position [nmi]:
10
Enter intruder's ground track [deg]:
180
Enter intruder's ground speed [knot]:
300
TCAS TAs: 62 [deg], 63 [deg], 64 [deg], 65 [deg], 66 [deg], 67 [deg],
68 [deg], 69 [deg], 70 [deg], 71 [deg], 72 [deg], 73 [deg], 74 [deg],
75 [deg], 76 [deg], 77 [deg], 78 [deg], 79 [deg], 80 [deg], 81 [deg],
82 [deg], 83 [deg], 84 [deg], 85 [deg], 86 [deg], 87 [deg], 88 [deg],
89 [deg], 90 [deg], 91 [deg], 92 [deg], 93 [deg], 94 [deg], 95 [deg],
96 [deg], 97 [deg], 98 [deg], 99 [deg], 100 [deg], 101 [deg], 102 [deg],
103 [deg], 104 [deg], 105 [deg], 106 [deg], 107 [deg], 108 [deg]
```

# Finally

```
top : THEORY
BEGIN

  IMPORTING TCAS_tau,         % Definition of tau and TCPA
            TCAS_TA_2D_core,  % 2-D core traffic alerting logic
            Units,            % Unit conversion functions
            TCAS_tables,      % TCAS threshold tables
            TCAS_TA_2D,       % TCAS TA 2-D logic
            TCAS_TA_2D_io

END top
```

```
$ proveit -a
Processing TutorialPVS.
Writing output to file ./TutorialPVS.summary
Grand Totals: 45 proofs, 45 attempted, 45 succeeded (46.75 s)
```

# Finally

```
top : THEORY
BEGIN

  IMPORTING TCAS_tau,         % Definition of tau and TCPA
            TCAS_TA_2D_core,  % 2-D core traffic alerting logic
            Units,            % Unit conversion functions
            TCAS_tables,      % TCAS threshold tables
            TCAS_TA_2D,       % TCAS TA 2-D logic
            TCAS_TA_2D_io

END top

$ proveit -a
Processing TutorialPVS.
Writing output to file ./TutorialPVS.summary
Grand Totals: 45 proofs, 45 attempted, 45 succeeded (46.75 s)
```

# Finally

```
top : THEORY
BEGIN

  IMPORTING TCAS_tau,         % Definition of tau and TCPA
            TCAS_TA_2D_core, % 2-D core traffic alerting logic
            Units,            % Unit conversion functions
            TCAS_tables,      % TCAS threshold tables
            TCAS_TA_2D,       % TCAS TA 2-D logic
            TCAS_TA_2D_io

END top
```

```
$ proveit -a
Processing TutorialPVS.
Writing output to file ./TutorialPVS.summary
Grand Totals: 45 proofs, 45 attempted, 45 succeeded (46.75 s)
```

# To Know More

- PVS: `https://pvs.csl.sri.com`
- VSCode-PVS: `https://shemesh.larc.nasa.gov/fm/VSCode-PVS`
- PVS @ NASA: `https://shemesh.larc.nasa.gov/fm/pvs`
- NASALib: `https://shemesh.larc.nasa.gov/fm/pvs/PVS-library`
- Formalization of Detect and Avoid: `https://shemesh.larc.nasa.gov/fm/DAIDALUS`