A  S Specification

Any alterations from the original S specification are indicated by “% mrd.”

% Converted to S / Fuss 2.51 by Michael Donat, Feb 1998
% minima.s
% Nancy A. Day
% 22 Jan 98

%sourcefile SeparationMinimaSpec.hpp

%linenum 723
%include table.s

%linenum 570
:flight;
%linenum 572
:location := Azores | BDA | CAN | Caribbean | IberianPeninsula
  | Iceland | Scandinavia | UnitedKingdom | USA ;

%linenum 576
:segment;
%linenum 577
:time == num;
%linenum 713
Min: ((num)set -> num);

% mrd

forall (a,b,S).
  Min (INSERT a (INSERT b S)) =
    (if a < b then Min (INSERT a S) else Min (INSERT b S));
forall a.Min {a} = a;

%linenum 115

Routes1 := {(USA,BDA);(CAN,BDA);(IberianPeninsula, Azores);
  (Iceland,Scandinavia);(Iceland, UnitedKingdom)};

%linenum 119

Routes2 := {(USA,Caribbean);(CAN,Caribbean);(BDA, Caribbean)};

%linenum 413
"WATRSSameDir LongSep" : (flight#flight) -> num;

%linenum 417
"WATRSOppDir NoLongSepPeriod" : (flight#flight) -> (time#time);

%linenum 467

Routes3 :=
{(USA,Caribbean); (CAN,Caribbean);
(BDA,Caribbean); (USA,BDA); (CAN,BDA)};

%linenum 587
IsSupersonic :(flight -> bool);
%linenum 588
IsTurbojet :(flight -> bool);
%linenum 589
HavePartOfRouteInMNPSAirspace : (flight -> bool);
%linenum 591
MeetMNPS : (flight -> bool);
%linenum 592
OnPublishedRoute : (flight -> bool);
%linenum 593
RouteDeparture : (flight -> location);
%linenum 594
RouteDestination : (flight -> location);
%linenum 596
MachTechniqueUsed : flight -> bool;
%linenum 603
FlightLevel : (flight -> num);
%linenum 604
InCruiseClimb : (flight -> bool);
%linenum 605
InWATRSAirspace : (flight -> bool);
%linenum 606
IsLevel : (flight -> bool);
%linenum 607
IsOutsideMNPSAirspace : (flight -> bool);
%linenum 609
IsWestOf60W : (flight -> bool);
%linenum 610
IsWestOf55W : (flight -> bool);
%linenum 611
"LatChange Per10DLong LessThan0Eq1" : flight -> bool;
%linenum 613
"LatChange Per10DLon LessThanOrEq2" :flight->bool;
%linenum 615
"LatChange Per10DLon LessThanOrEq3" :flight->bool;
%linenum 617
LateralPositionInDegrees : (flight -> num);
%linenum 619
LateralPositionInMiles : (flight -> num);
%linenum 621
Mach : (flight->num);
%linenum 622
RouteSegment : (flight -> segment);
%linenum 623
"RouteSegment Degrees" : (flight -> num);
%linenum 624
TimeAtPosition : (flight->time);
%linenum 629
SameType : ((flight # flight) -> bool);
%linenum 635
SameMachNumber : ((flight # flight) -> bool);
%linenum 637
FirstAircraft : ((flight # flight) -> flight);
%linenum 639
ept : ((flight # flight)->time);
%linenum 640
"SameOr Diverging Tracks" : ((flight # flight) -> bool);
%linenum 642
SecondAircraft : ((flight # flight) -> flight);
%linenum 647
ReportedOverCommonPoint : ((flight # flight) -> bool);
%linenum 649
"Appropriate TimeSep AtCommon Point" : ((flight # flight) -> bool);
%linenum 651
EnterWATRSAirspaceAtSomeTime : (flight -> bool);
%linenum 657
AngularDifferenceGreaterThan90Degrees : (segment # segment)->bool;
%linenum 668
% mrd
startTime (s:time, e:time) := s;
%linenum 672
% mrd
endTime (s:time, e:time) := e;
%linenum 677
MinEarliestTime : (time # time)set -> time;
%linenum 682
MaxLatestTime : (time # time)set -> time;

forall (p,x,S).MaxLatestTime (INSERT p (INSERT x S)) =
  (let a := EndTime p in
  let b := MaxLatestTime (INSERT x S) in
  if a > b then a else b);

forall p.MaxLatestTime (INSERT p EMPTY) = EndTime p;

forall (p,x,S).MinEarliestTime (INSERT p (INSERT x S)) =
  (let a := StartTime p in
  let b := MinEarliestTime (INSERT x S) in
  if a < b then a else b);

forall p.MinEarliestTime (INSERT p EMPTY) = StartTime p;

VerticalSeparationRequired (A,B) := Table
  [Row (FlightLevel (A)) [[(x.x <= 280); DC; (x.x>450);(x.x>450)]];]
  [Row (FlightLevel (B)) [DC;(x.x <= 280);(x.x > 450);(x.x>450)]];]
  [Row (IsSupersonic (A)) [DC;DC;TRUE;DC];]
  [Row (IsSupersonic (B)) [DC;DC;TRUE] ]
  [1000;1000;4000;4000;2000];]

IsOnRoute (R:(location#location)set) (X:flight) :=
  ( (RouteDeparture (X), RouteDestination (X) In R) OR
    (RouteDestination (X), RouteDeparture (X) In R));

forall (a,b,c,d).((a,b) = (c,d)) = (a = c) /
  (b = d);

FlightLevelAbove275 (X:flight) := FlightLevel X > 275;
"Lateral Separation Required In Degrees" (A,B) := Table
[Row (A10f [A;B] IsOutsideMNPSAirspace) [TRUE;TRUE;DC;DC];
Row (A10f [A;B] (IsOnRoute (Routes1))) [TRUE;DC;DC;DC];
Row (A10f [A;B] (IsOnRoute (Routes2))) [DC;TRUE;DC;DC];
Row (A10f [A;B] IsWestOf55W) [DC;TRUE;DC;DC];
Row (A10f [A;B] IsSupersonic) [DC;DC;TRUE;DC];
Row (A10f [A;B] FlightLevelAbove275) [DC;DC;TRUE;DC];
Row (A10f [A;B] MeetMNPS) [DC;DC;DC;TRUE];
Row (A10f [A;B] HavePartOfRouteInMNPSAirspace) [DC;DC;DC;TRUE]]
[1.5;1.5;1;1;2];

"Lateral Separation Required In Miles" (A,B) := Table
[Row (A10f [A;B] IsOutsideMNPSAirspace) [TRUE;TRUE;DC;DC];
Row (A10f [A;B] (IsOnRoute (Routes1))) [TRUE;DC;DC;DC];
Row (A10f [A;B] (IsOnRoute (Routes2))) [DC;TRUE;DC;DC];
Row (A10f [A;B] IsWestOf55W) [DC;TRUE;DC;DC];
Row (A10f [A;B] IsSupersonic) [DC;DC;TRUE;DC];
Row (A10f [A;B] FlightLevelAbove275) [DC;DC;TRUE;DC];
Row (A10f [A;B] MeetMNPS) [DC;DC;DC;TRUE];
Row (A10f [A;B] HavePartOfRouteInMNPSAirspace) [DC;DC;DC;TRUE]]
[90;90;60;60;120];

LatitudeEquivalent (A,B) := PredicateTable
[Row ("RouteSegment Degrees" A)
[(\x.x<=58);DC;((x.(68<x) AND (x<70));DC;((\x.(70<=x) AND (x<=80));DC;
Row ("RouteSegment Degrees" B)
[DC;((x.x<=58);DC;((x.(68<x) AND (x<70));DC;((\x.(70<=x) AND (x<=80))];
Row (A10f [A;B] "LatChange Per10DLong LessThanOrEq3"
[TRUE;TRUE;DC;DC;DC];
Row (A10f [A;B] "LatChange Per10DLong LessThanOrEq2"
[DC;DC;TRUE;TRUE;DC;DC];
Row (A10f [A;B] "LatChange Per10DLong LessThanOrEq1"
[DC;DC;DC;DC;TRUE;TRUE]];
SSSubcondition(A,B) := PredicateTable
   [Row (AllOf [A;B] IsLevel) [TRUE;DC];
    Row (SameMachNumber (A,B)) [TRUE;DC];
    Row (SameType(A,B)) [DC;TRUE];
    Row (AllOf [A;B] InCruiseClimb) [DC;TRUE];]

UnionOfRange (periods) :=
   (MinEarliestTime (periods), MaxLatestTime (periods));

MNPSCondition(A,B) :=
   (AllOf [A;B] MeetMNPS) AND
   (AllOf [A;B] HavePartOfRouteInMNPSAirspace);

"MNPSOppDir NoLongSepPeriod"(A,B) := "ssOppDir NoLongSepPeriod"(A,B);

MNPSSameDir LongSep (A,B) := Table
   [Row ("Appropriate TimeSep AtCommon Point" (A,B))
   [TRUE;TRUE;TRUE;TRUE];
    Row ("SameOr Diverging Tracks" (A,B)) [TRUE;TRUE;TRUE;TRUE];
    Row (Mach (FirstAircraft (A,B)) - Mach (SecondAircraft (A,B)))[
   (\x. (x>0.06));(\x. ((0.06>=x) AND (x>0.05)));(\x. ((0.05>=x) AND (x>0.04)));
   (\x. ((0.04>=x) AND (x>0.03)));(\x. ((0.03>=x) AND (x>0.02))]]
   [5;6;7;8;9;10];

WATRSCcondition(A,B) := PredicateTable
   [Row (AllOf [A;B] EnterWATRSAirspaceAtSomeTime) [TRUE;TRUE];
    Row (AllOf [A;B] IsWestOf60W) [TRUE;DC];
    Row (AllOf [A;B] InWATRSAirspace) [DC;TRUE];
    Row (AllOf [A;B] MachTechniqueUsed) [TRUE;TRUE];
    Row (AllOf [A;B] OnPublishedRoute) [TRUE;TRUE];
    Row ("SameOr Diverging Tracks" (A,B)) [TRUE;TRUE];]
"genOppDir NoLongSep Period"(A,B) := "MNPSOppDir NoLongSepPeriod"(A,B);

"genSameDir LongSep" (A,B) := Table
[Row ("SameOr Diverging Tracks" (A,B)) [TRUE;TRUE;TRUE];
 Row (AllOf [A;B] MachTechniqueUsed ) [FALSE;TRUE;TRUE];
 Row (AtLeastOneOf [A;B] InCruiseClimb ) [FALSE;FALSE;FALSE];
 Row (ReportedOverCommonPoint (A,B)) [TRUE;DC;DC];
 Row ("Appropriate TimeSep AtCommon Point" (A,B)) [DC;TRUE;TRUE];
 Row (Mach (FirstAircraft(A,B)) - Mach (SecondAircraft(A,B)))
 [DC;\(\{x. (x>0.6));(\{x. (0.6>=x) AND (x>0.3)\}\)]
[15;5;10;20];

"otherOppDir NoLongSepPeriod" (A,B) := "genOppDir NoLongSep Period"(A,B);

otherSameDirLongSep (A,B) := Table
[Row (ReportedOverCommonPoint(A,B)) [TRUE;DC];
 Row ("SameOr Diverging Tracks"(A,B)) [TRUE;DC];
 Row (AllOf [A;B] (IsOnRoute Routes3)) [DC(TRUE)];
[15;20;30];

env1 :=
(forall (A:flight). NOT (IsLevel (A) AND InCruiseClimb (A)))
AND
(forall (A:flight).NOT (IsOnRoute (Routes1) (A) AND IsOnRoute (Routes2) (A)));

forall A.MutEx [IsLevel (A); InCruiseClimb (A)];
forall A.MutEx [IsOnRoute (Routes1) (A); IsOnRoute (Routes2) (A)];

env2 :=
(forall (A:flight) (B:flight).
  ReportedOverCommonPoint(A,B) = ReportedOverCommonPoint(B,A))
AND
(forall (A:flight) (B:flight).
  SameMachNumber(A,B) = SameMachNumber(B,A))
AND
(forall (A:flight) (B:flight).
  SameType(A,B) = SameType(B,A))
AND
(forall (A:flight) (B:flight).
  "SameOr Diverging Tracks"(A,B) = "SameOr Diverging Tracks"(B,A))
AND
(forall (A:flight) (B:flight).
  "Appropriate TimeSep AtCommon Point"(A,B) =
   "Appropriate TimeSep AtCommon Point"(B,A));

% mrd env2
forall (A,B).Subsm [ReportedOverCommonPoint(A,B); ReportedOverCommonPoint(B,A)]; 
forall (A,B).Subsm [SameMachNumber(A,B); SameMachNumber(B,A)]; 
forall (A,B).Subsm [SameType(A,B); SameType(B,A)];
forall (A,B).
  Subsm ["SameOr Diverging Tracks"(A,B); "SameOr Diverging Tracks"(B,A)];
forall (A,B).Subsm [
  "Appropriate TimeSep AtCommon Point"(A,B);
  "Appropriate TimeSep AtCommon Point"(B,A)];

%linenum 523

env3 :=
(fforall A.
  if "LatChange Per10DLon LessThanOrEq2" (A)
    then "LatChange Per10DLon LessThanOrEq3" (A))
AND
(fforall A.
  if "LatChange Per10DLon LessThanOrEq1" (A)
    then "LatChange Per10DLon LessThanOrEq2" (A))
AND
(fforall A.
  if "LatChange Per10DLon LessThanOrEq1" (A)
    then "LatChange Per10DLon LessThanOrEq3" (A));

% mrd env3
forall A.\text{Subsm} [ \\
"\text{LatChange Per10DLon} \text{g LessThanOrEq}3" \ A; \\
"\text{LatChange Per10DLon} \text{g LessThanOrEq}2" \ A; \\
"\text{LatChange Per10DLon} \text{g LessThanOrEq}1" \ A];

\%linenum 540

\text{env} := \text{env1 AND env2 AND env3};

\%linenum 254

\text{ssSameDirLongSep}(A, B) := \text{Table} \\
[ \text{Row (ssSubcondition}(A, B)) \ [\text{TRUE};\text{TRUE}] ; \\
\text{Row ("SameOr Diverging Tracks"}(A, B)) \ [\text{TRUE};\text{TRUE}] ; \\
\text{Row (ReportedOverCommonPoint}(A, B)) \ [\text{TRUE};\text{DC}] ; \\
\text{Row ("Appropriate TimeSep AtCommon Point"}(A, B)) \ [\text{DC};\text{TRUE}] ] \\
[10;10;15];

\%linenum 306

\text{MinAll}(A, B) := \\
\text{Min} \\
\{ \\
"\text{MNPSSameDir LongSep}(A, B); \\
"\text{WATRSSameDir LongSep}(A, B); \\
"\text{genSameDir LongSep}(A, B)\};

\%linenum 341

\text{UnionAll} (A, B) := \\
\text{let periods :=} \\
\{"\text{MNPSOppDir NoLongSepPeriod"}(A, B); \\
"\text{WATRSOppDir NoLongSepPeriod"}(A, B); \\
"\text{genOppDir NoLongSep Period"}(A, B)\} \text{in} \\
(\text{MinEarliestTime (periods)}, \text{MaxLatestTime (periods)});

\%linenum 291

"\text{turbojetSameDir LongSep"}(A, B) := \text{Table} \\
[ \text{Row (MNPSCondition} (A, B)) \ [\text{TRUE};\text{FALSE};\text{TRUE};\text{FALSE}] ; \\
\text{Row (WATRSCondition} (A, B)) \ [\text{TRUE};\text{TRUE};\text{FALSE};\text{FALSE}] ] \\
[\text{MinAll} (A, B); \\
\text{Min} \{ "\text{WATRSSameDir LongSep"} (A, B);
WithinOppDirNoLongSepPeriod refers to the time at which separation is considered. Since all other predicates are assumed to refer to this same time, the variable "separation check time" is used to represent it. This is more appealing than adding a time parameter to each of the other predicates. This removes time as a parameter from AreSeparated.

If it were necessary to consider separation at two different times,
then each of the predicates would need to be parameterized by the
time at which separation was being considered.

*/

"separation check time" : time;

WithinOppDirNoLongSepPeriod(A:flight,B:flight) :=
  let t := "separation check time" in
  let timePeriod := "OppDir NoLongSepPeriod"(A,B) in
  (StartTime(timePeriod) <= t) AND (t <= EndTime(timePeriod));

/* mrd */

/* A and B are vertically separated based on flight level */
VerticallySeparated(A,B) :=
  ABS(FlightLevel A - FlightLevel B) > VerticalSeparationRequired(A,B);

/* A and B are laterally separated based on either position in degrees
   of latitude or position in miles */
LaterallySeparated(A,B) :=
  if (LatitudeEquivalent(A,B))
    then
      (ABS(LateralPositionInDegrees A - LateralPositionInDegrees B) >
       "LateralSeparation RequiredInDegrees" (A,B))
    else
      (ABS(LateralPositionInMiles A - LateralPositionInMiles B) >
       "LateralSeparation RequiredInMiles" (A,B));

/* A and B are longitudinally separated based on time
   depending on whether the two flights are in the approximate
   same or opposite direction */
LongitudinallySeparated(A,B) :=
  if (AngularDifferenceGreaterThan90Degrees
       (RouteSegment A, RouteSegment B))
    then  /* opposite direction */
      NOT (WithinOppDirNoLongSepPeriod(A,B))
else  /* same direction */
    ABS(TimeAtPosition A - TimeAtPosition B) >
    LongSameDirSepRequired(A,B);

AreSeparated(A:flight,B:flight) :=
    VerticallySeparated(A,B) OR
    LaterallySeparated(A,B) OR
    LongitudinallySeparated(A,B);

%%% condition "ssOppDir NoLongSepPeriod"
%%% condition ssSubcondition
%%% condition "genSameDir LongSep"
%%% condition ssSameDirLongSep
%%% condition "turbojetSameDir LongSep"
%%% condition "turbojetOppDir NoLongSepPeriod"

/* Misc simplification */

forall (a,b).a < a - b = b < 0;
forall (a,b).a - b < a = b > 0;
forall (a,b).a <= a - b = b <= 0;
forall (a,b).a - b <= a = b >= 0;
forall (a,b,c).a + b < a + c = b < c;
forall (a,b,c).a + b <= a + c = b <= c;

/* Condition dependencies for differentiated test frames */

forall a.MutEx [
    "LatChange Per10DLong LessThanOrEq2" a);
    "LatChange Per10DLong LessThanOrEq3" a];

forall a.MutEx [
    "LatChange Per10DLong LessThanOrEq1" a);
    "LatChange Per10DLong LessThanOrEq3" a];

forall a.MutEx [
    "LatChange Per10DLong LessThanOrEq1" a);
    "LatChange Per10DLong LessThanOrEq2" a];

/* - dependencies required due to iteration */

forall (A,B).MutEx [
    "separation check time" < ept (A, B);
EndTime ("turbojetOppDir NoLongSepPeriod" (A, B)) < "separation check time"

forall (A,B,x). x > 0 => MutEx [
  "separation check time" < ept (A, B) - x;
  EndTime ("turbojetOppDir NoLongSepPeriod" (A, B)) < "separation check time"
]

forall (A,B). MutEx [
  "separation check time"
    < StartTime ("turbojetOppDir NoLongSepPeriod" (A, B));
    ept (A, B) <= "separation check time"
]

forall (A,B,x). x > 0 => MutEx [
  "separation check time"
    < StartTime ("turbojetOppDir NoLongSepPeriod" (A, B));
    ept (A, B) + x <= "separation check time"
]

forall (A,B). (ReportedOverCommonPoint (A, B)) => MutEx [
  "separation check time" <
    StartTime ("turbojetOppDir NoLongSepPeriod" (A, B));
    ept (A, B) - 15 <= "separation check time"
]