Modular specification of real-time systems

Inal, Recep

Published in:
Proceedings of the Sixth Euromicro Workshop on Real-Time Systems

Link to article, DOI:
10.1109/EMWRTS.1994.336871

Publication date:
1994

Document Version
Publisher's PDF, also known as Version of record

Modular Specification of Real-Time Systems *

Recep Inal
Department of Computer Science
Technical University of Denmark, bldg. 344
DK 2800 Lyngby, Denmark
Email: rei@id.dth.dk

Abstract

Duration Calculus, a real-time interval logic, has been embedded in the Z specification language to provide a notation for real-time systems that combines the modularisation and abstraction facilities of Z with a logic suitable for reasoning about real-time properties [4, 2]. In this article the notation is presented through a top-level specification of requirements for a simple Air Traffic Monitoring system, and reasoning is illustrated by a refinement towards a design.

Keywords: real-time systems, requirements, formal specifications, refinement, design.

1 Introduction

A purpose of formal specifications is to contribute to concise and understandable documentation of requirements, assumptions and design. Furthermore they shall allow a designer to reason about correctness of a design with respect to requirements under given assumptions. Previous work within the Provably Correct Systems project [1] has demonstrated use of Duration Calculus, a real-time interval logic as a notation for concise specification and reasoning about real-time systems [9, 3, 5]. In that work, the notation is used in the usual mathematical fashion with implicit declarations and modularisation through textual subdivision. This paper illustrates an application of a new development, where the Duration Calculus is embedded in the widely used Z language [11] which provides a precise explicit notation for modular specifications suitable for larger scale developments.

The timed Z notation is illustrated by means of a case study which is based on an air traffic monitoring problem posed by the University of Stirling[7]. The paper is organized so it documents the development stages:

1. Definition of a system model, and specification of requirements.
2. Specification of a control design for the system and assumptions about the environment.
3. Verification of the control design by proving that the design refines the requirements under the assumptions.

Although the documentation is presented in a top-down fashion, it must be noted that development of a sufficiently abstract system model and a correct design is an iterative process.

In the conclusion we report experience with a proof assistant tool encoded in PVS [8, 10] for mechanical verification. In order to give a feeling of how proofs in the Duration Calculus are structured the detailed verifications are given in an appendix.

2 Air Traffic Monitoring System

The air traffic monitoring system monitors the position of civil aircraft in the vicinity of an airport. Aircraft may be landing at, departing from or crossing over the airport. The aims of the system are:

1. to ensure aircraft do not risk mid-air collision by coming too close to each other,
2. to organize landing and departings of aircraft, and
3. to collect position information from radar

The components of a system model are: an *Airspace* database containing the aircraft, a *Radar* that measures the positions of aircraft, an *Alert* mechanism that alerts aircraft in collision risk, and finally one or more *Runways* for the landing and departures of aircraft. The task of a software design for the system is to take care of the correct interplay between the components.

The tasks of the system motivates a modular composition of the overall system. To keep the presentation short, we will focus on the collision avoidance part of the system, and hence describe the interplay between *Airspace* and *Alert* components. This subsystem will be denoted as CAS, Collision Avoidance System. It should be noted that the developed model is kept deliberately simple. For instance the velocity of aircraft is not modeled, but this can easily be added, and is used in the larger model presented in [6].

*This work is partially funded by the Commission of the European Communities (CEC) under the ESPRIT programme in the field of Basic Research Project No. 7071: ProCoS: Provably Correct Systems".

3 System Model

In this section a system model of the CAS subsystem is derived in order to capture the safety requirement. First we introduce some design parameters:

$$v_{max}, r, R : \mathbb{R}$$
$$T, T_{al}, T_{coll}, T_{init} : Time$$

These parameters will be explained below.

Air space:
A state is needed to track all aircraft in the controlled air space. This state, which is a real-time database recording all aircraft positions, will be named Pos. It is declared in a schema.

AIRSPACE
$$Pos : ID \rightarrow (\textbf{State}\ POSITION)$$

Since the notion of time is implicit in our notation we use the keyword **State** to denote time varying states. A variable with this keyword is interpreted as a function of time.

The database is a function from aircraft-identifiers, $[ID]$, to information about the aircraft, in this simple case a position, which changes over time. The fact that no aircraft can be in different places in same time is thus captured by the model variable. The positions are Cartesian coordinates with the origin being the location of the radar.

$$POSITION \mathrel{\hat{=}} [\ x : \mathbb{R};\ y : \mathbb{R};\ z : \mathbb{R}\ ]$$

Distance between positions is computed by a function $dist$. We define first a predicate $Close$, which holds if the distance between two aircraft becomes less than some input parameter $x?$:

Close
$$AIRSPACE$$
$$x? : \mathbb{R}$$
$$\exists\, id_1, id_2 : \mathrm{dom}\ Pos \bullet \lceil Pos(id_1) = p_1 \wedge Pos(id_2) = p_2 \rceil \wedge id_1 \neq id_2 \wedge dist(p_1, p_2) < x?$$

Note that in Z all free variables are universally quantified.

Aircraft are surrounded and protected by a sphere, with radius, $r \geq 0$, that no other aircraft is permitted to enter. For simplicity we assume all aircraft have the same speed, and hence the same protection sphere. A violation of this sphere, i.e if the distance between two aircraft becomes less than $2r$ is tantamount to a collision. Hence the predicate $Coll$:

$$Coll \mathrel{\hat{=}} Close[2r/x?]$$

The predicate is defined by renaming of the input parameter $x?$ to r.

3.1 Requirements

We consider a single safety requirement, REQ, namely the fact that no collisions may take place in the controlled air space.

$$REQ \mathrel{\hat{=}} \Box(\neg Coll)$$

The modal operator, $\Box$, means for any interval of time. The formula says that for any interval of time the predicate $\neg Coll$ holds.

4 Control System Design

In this section a refinement of the system model towards a design is presented. The design consists of some assumptions that should be satisfied by the environment and a controller in terms of a finite state automaton.

4.1 Assumptions

To detect collision risks a bigger sphere with radius, $R > r$, surrounding each aircraft is needed. A collision risk appears when the distance between two arbitrary aircraft becomes less than $2R$. Figure 1 illustrates the idea.

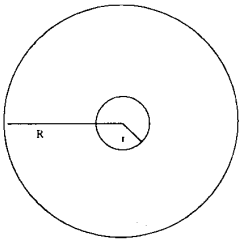

Figure 1: Protection spheres

We will also assume a general maximum velocity, $v_{max} > 0$, for all aircraft in the air space. The worst collision time, T_{coll} occurs when two aircraft, separated by a distance of $2R$, move against each other with this maximum velocity. This time is given by the equation

$$T_{coll} = \frac{R-r}{v_{max}}$$

Furthermore we will assume that the positions only change after T time units, i.e. a measured position is stable for time T. In order to be able to detect collision risks in time we require that the aircraft do not move too fast. These two assumptions are expressed as:

ASM_0
$$AIRSPACE$$
$$\Box(\lceil Pos(id) \neq p_1 \rceil\ ;\ \lceil Pos(id) = p_1 \rceil\ ;\ \lceil Pos(id) \neq p_1 \rceil \Rightarrow \ell > T)$$
$$\Box(\lceil Pos(id) = p_1 \rceil\ ;\ \textbf{true}\ ;\ \lceil Pos(id) = p_2 \rceil \Rightarrow dist(p_1, p_2) \leq v_{max} \cdot (\ell + T))$$

These formula use the chop operator ';' to divide an interval into contiguous subintervals. The first formula considers an interval, where a given aircraft *id* change to position p_1 and then away again. For any interval of this form the length, ℓ, is greater than T. The second formula states that for any interval consisting of an initial subinterval where an aircraft with *id* has position p_1 followed by a unspecified subinterval, **true**, and followed by a third subinterval in which the same aircraft has now position p_2 then the flown distance is less than $v_{max} \cdot (\ell + T)$.

From ASM_0 we are able to set an upper bound on the sampling time T. First we need a predicate, *AllSafe*, which holds if all aircraft are safely separated:

$$AllSafe \mathrel{\hat{=}} \neg Close[2R/x?]$$

Fact1: A potential collision is detectable if all aircraft are safe before a collision and if $T \leq T_{coll}$.

$$\Box(AllSafe\ ;\ \mathbf{true}\ ;\ Coll \Rightarrow \ell \geq T_{coll} - T)$$

Fact2: A collision risk may be missed by the system if $T > T_{coll}$.

$$\Box(AllSafe\ ;\ Coll \Rightarrow T > T_{coll})$$

Proofs of these facts are given in Appendix B. As a consequence of these lemmas we will assume that $T_{coll} \geq T$.

Alert:
To ensure safety, the system must alert the pilots of aircraft being in a collision risk, i.e. whenever two aircraft come too close to each other. The pilots of aircraft in collision are then alerted via an air traffic controller. We introduce a state variable, *Alert*, a set of identifiers, for aircraft in collision risk. The states of the CAS system are now collected in the schema:

CAS_STATES
$AIRSPACE$
$Alert : \mathbf{State}\ \mathbb{P}\ ID$

The interplay between *Alert* and the pilots is not specified in this work, but instead we assume that an intelligent plan for avoiding the collisions is produced by the air traffic controller, and that the pilots follow the plan and are out of the danger zone within T_{al} time units. Finally we will assume that in an initial interval of length T_{init} all aircraft are safely separated. The above assumptions are now formalised in a schema:

ASM$_1$
CAS_STATES
ASM_0

$R > r$
$T_{coll} \geq T$
$\lceil InDanger \subseteq Alert \rceil \overset{T_{al}}{\rightarrow} \lceil InDanger = \{\} \rceil$
$0 < \ell \leq T_{init} \Rightarrow AllSafe$

The third assumption is expressed by use of the timed progress operator:

$$D_1 \overset{T}{\rightarrow} D_2 \mathrel{\hat{=}} \Box((D_1 \wedge \ell = T)\ ;\ \ell > 0 \Rightarrow \ell = T\ ;\ D_2\ ;\ \mathbf{true})$$

The formula states that if a formula D_1 holds for time T on an interval then formula D_2 holds initially on a succeeding non-point interval.

The function *InDanger* finds all aircraft being in a collision risk.

InDanger
$AIRSPACE$
$ds! : \mathbf{State}\ \mathbb{P}\ ID$

$ds! = (\lambda\, t : Time \bullet$
$\{id \mid id : \mathrm{dom}\, Pos \wedge$
$\exists\, id_1 : \mathrm{dom}\, Pos \bullet id \neq id_1 \wedge$
$dist(Pos(id)(t), Pos(id_1)(t)) < 2R\})$

The output parameter, *ds*!, records the aircraft having a collision risk as a function of time.

4.2 Controller Design

The controller design is based on a finite state automaton. We introduce a new type *PHASES* to represent the phases of the automaton.

$$[PHASES]$$

The control automaton consists of three different phases. These are introduced as:

$Init, Idle, Act : PHASES$

$Init \neq Idle \wedge Idle \neq Act \wedge Act \neq Init$

The *Init* phase describes the initial behaviour of the controller. In the *Idle* phase the automaton does nothing, and in the *Act* phase the automaton performs actions for avoiding possible collisions. The abbreviations $\lceil \mathsf{Init} \rceil$, $\lceil \mathsf{Idle} \rceil$ and $\lceil \mathsf{Act} \rceil$ are in the following used to denote the current phase of the automaton. The phases of the automaton and the transitions between them are depicted in figure 2.

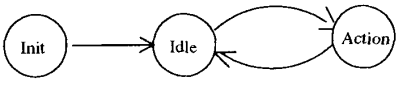

Figure 2: automaton

We need two new constants: ϵ and δ. The parameter ϵ is used to specify a latency time for the system to react and move from one phase to another. The parameter δ is used for a latency time for the system to enable the alert mechanism.

$\epsilon, \delta : Time$

$T_{coll} \geq T + T_{al} + \delta + \epsilon$

We decide that the automaton starts in the *Init* phase, and that the *Init* phase is followed by the *Idle* phase within T_{init} time units.

$$INITIAL \hat{=} (\lceil\ \rceil \rightarrow \lceil \mathsf{Init} \rceil) \wedge (\lceil \mathsf{Init} \rceil \overset{T_{init}}{\rightarrow} \lceil \mathsf{Idle} \rceil)$$

The next schema states that the *Idle* phase is followed by the *Act* phase.

$$TRANS \hat{=} \lceil \mathsf{Idle} \rceil \rightarrow \lceil \mathsf{Act} \rceil$$

The predicate *Danger*, defined as

$$Danger \hat{=} Close[2R/x?]$$

determines when a transition is allowed to be performed. The predicate states simply if there is risk for a collision. The enabled transitions are specified in the schema *ENAB_TRANS*.

ENAB_TRANS

AIRSPACE

$$\lceil \mathsf{Idle} \rceil \wedge Danger \overset{\epsilon}{\rightarrow} \lceil \mathsf{Act} \rceil$$

$$\lceil \mathsf{Act} \rceil \wedge \neg Danger \overset{\epsilon}{\rightarrow} \lceil \mathsf{Idle} \rceil$$

The next schema, *CON_STABILITY*, describe the conditions needed to maintain a phase.

CON_STABILITY

AIRSPACE

$$\lceil \mathsf{Idle} \rceil \wedge \neg Danger \rightarrow \lceil \mathsf{Idle} \rceil$$

$$\lceil \mathsf{Act} \rceil \wedge Danger \rightarrow \lceil \mathsf{Act} \rceil$$

The actuator part of the system, i.e. the alert mechanism, is activated in the *Act* phase. This decision is specified as:

ACTIONS

CAS_STATES

$$\lceil \mathsf{Act} \rceil \overset{\delta}{\rightarrow} \lceil InDanger \subseteq Alert \rceil$$

The total specification of the controller design is given by a collection of schemas:

$$DES_1 \hat{=} INITIAL \wedge TRANS \wedge ENAB\_TRANS \wedge ACTIONS \wedge CON\_STABILITY$$

As a final specification we introduce a schema for the control design of the CAS subsystem:

$$CAS_1 \hat{=} DES_1 \wedge ASM_1$$

The proof obligation is then to show the correctness of the control design, i.e.

$$CAS_1 \Rightarrow REQ$$

which is verified in Appendix A.

5 Conclusion

We have illustrated an application of a Timed-Z notation comprising of an embedding of the Duration Calculus into the Z specification language. The advantages of this is twofold: besides specifying and reasoning about timing properties of real-time systems, it provides also composite modular specifications. The latter is a necessity when dealing with complex systems. The example of the specification of a subsystem of the Air Traffic Monitoring System illustrates this thesis. The other subsystems have been specified in a similar manner as done here [6].

Another advantage of the presented approach is the possibility for application of the existing software tools. The presented specification and verification has been checked by the DC/PC tool. This has resulted in detection of some missing details. For instance the stability of the measured positions was not captured in the initial specification, but the proof assistant insisted on the necessity of this fact. Thus for large scale development of real-time systems the assistance of tools for type checking and mechanical verification is indispensable. This will not reduce design time, but it will certainly liberate the designer from the tedious parts of the verification, and ensure the correctness of the design.

Acknowledgements

I am very grateful to Anders P. Ravn for his inspiring and indeed very valuable comments and ideas on this case study. Thanks to Kirsten M. Hansen and Jens U. Skakkebæk for their helpful comments and suggestions.

References

[1] D. Bjørner. A ProCoS Project Description. ESPRIT BRA 3104, EATCS Bull. no. 39, Oct. 1989.

[2] S.M. Brien, M. Engel, He Jifeng, H. Rischel, and A.P. Ravn. *Z Description of Duration Calculus.* Technical report, Draft. OU HJF 12/2

[3] J. Bowen, M. Franzle, E.R. Rüdiger, and A.P. Ravn. Developing Correct Systems. *Proc. 5th Euromicro Workshop on Real-Time Systems*, pages 176–187, June 1993.

[4] M. Engel. Specifying Real-Time Systems with Z and the Duration Calculus. To appear in *ZUW'94*

[5] K.M. Hansen, H.Rischel, and A.P. Ravn. Specifying and Verifying Requirements of Real-Time Systems. *IEEE Transactions on Software Engineering*, Vol 19(1):41–55, 1993.

[6] R. Inal. *Specifying and Verifying Real-Time Systems.* Master's thesis, ID/DTH/RI, 1993.

[7] K. Jackson. Aircraft Traffic Control. Technical report, University of Stirling, 1992.

[8] S. Owre, N. Shankar, and J.M. Rushby. *User Guide for the PVS Specification and Verification System, Language and Proof Checker.* Technical report, CSL, SRI Intl., October 1993.

[9] J.U. Skakkebæk, H. Rischel, A.P. Ravn, and Z. Chaochen. Specification of Embedded Real-Time Systems. *Proc. 4th Euromicro Workshop on Real-Time Systems*, pages 116–121, June 1992.

[10] J.U. Skakkebæk, and N. Shankar. Towards a Duration Calculus Proof Assistant in PVS. Technical Report ID/DTH/JUS 5/1, Lyngby, Denmark March 1994. Paper version submitted for publication.

[11] J.M. Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall, 2nd edition, 1992.

A Appendix A

Our single safety requirement is proved by contradiction. Assume

$$\neg REQ \Leftrightarrow \neg(\Box(\neg Coll)) \Leftrightarrow \Diamond Coll$$

The formula, $\Diamond Coll$, is split into three cases:

Case a : $Coll$; **true**
Case b : $AllSafe$; $Coll$
Case c : $AllSafe$; $Danger$; $Coll$

which corresponds to occurrence of a collision, respectively, at the beginning of the observation interval, immediately after a situation, where all aircraft are safely separated, and immediately after a situation, where a collision risk is observed.

Case b can not occur, because our design does not, by $Fact2$ and ASM_1, allow such big changes of positions. We examine now the first case. *INITIAL* decomposes to $\lceil\ \rceil$ or $\lceil \mathsf{Init} \rceil$; **true**. It is easy to see that $\lceil\ \rceil \wedge Coll$; **true** $\Rightarrow$ **false**. The rest of the analysis of the first case gives:

$Coll$; **true** $\wedge$ $\lceil \mathsf{Init} \rceil$; **true**
$\Rightarrow \{INITIAL\}$
$Coll$; **true** $\wedge$ $(\ell \leq T_{init} \wedge \lceil \mathsf{Init} \rceil \vee$
$\ell = T_{init}$; $\lceil \mathsf{Idle} \rceil$; $true)$
$\Rightarrow \{PLogic,\ ASM_1\}$
$Coll$; **true** $\wedge$ $AllSafe \vee$
$Coll$; **true** $\wedge$ $AllSafe$; $\lceil \mathsf{Idle} \rceil$; **true**
$\Rightarrow \{DC\}$
$\Diamond(Coll \wedge AllSafe)$
$\Rightarrow \{R > r\}$
$\Diamond$**false**
$\Rightarrow \{DC\}$
false

Before starting on the analysis of Case c we state a lemma:

Lemma1: If the automaton is in the *Idle* phase while there is danger for collision then the danger will be avoided within $\epsilon + \delta + T_{al}$ time units.

$$\Box((\lceil \mathsf{Idle} \rceil\ ;\ \mathbf{true} \wedge Danger) \Rightarrow (\ell \leq \epsilon + \delta + T_{al} \vee \ell = \epsilon + \delta + T_{al}\ ;\ AllSafe))$$

Proof:

$\lceil \mathsf{Idle} \rceil$; **true** $\wedge$ $Danger$
$\Rightarrow \{ENAB\_TRANS\}$
$\ell \leq \epsilon \wedge \lceil \mathsf{Idle} \rceil \wedge Danger \vee$
$\ell = \epsilon$; $(\lceil \mathsf{Act} \rceil$; **true** $\wedge$ $Danger)$
$\Rightarrow \{\ell \geq \epsilon\}$
$\ell = \epsilon$; $(\lceil \mathsf{Act} \rceil$; **true** $\wedge$ $Danger)$
$\Rightarrow \{ACTIONS,\ \ell \geq \epsilon + \delta\}$
$\ell = \epsilon + \delta$; $(\lceil \mathsf{Act} \rceil$; **true** $\wedge$ $Danger \wedge$
$\lceil InDanger \subseteq Alert \rceil)$
$\Rightarrow \{ASM_1,\ CON\_STABILITY,\ \ell \geq \epsilon + \delta + T_{al}\}$
$\ell = \epsilon + \delta + T_{al}$; $(\lceil \mathsf{Act} \rceil$; **true** $\wedge$ $\lceil InDanger = \emptyset \rceil)$
$\Rightarrow \{Definitions\}$
$\ell = \epsilon + \delta + T_{al}$; $AllSafe$

Case is split into three more cases, each depending on the current phase of the automaton. We will go through the case, where the current phase is *Idle*:

$\lceil \mathsf{Idle} \rceil$; **true** $\wedge$ $(AllSafe$; $Danger$; $Coll)$
$\Rightarrow \{CON\_STABILITY\}$
$AllSafe$; $(\lceil \mathsf{Idle} \rceil$; **true** $\wedge$ $Danger$; $Coll)$
$\Rightarrow \{Lemma1\}$
$AllSafe$; $(\ell \leq \epsilon + \delta + T_{al} \vee$
$\ell = \epsilon + \delta + T_{al}$; $AllSafe)$; $Coll)$
$\Rightarrow \{DC\}$
$AllSafe$; $\ell \leq \epsilon + \delta + T_{al}$; $Coll \vee$
$AllSafe$; $\ell = \epsilon + \delta + T_{al}$; $AllSafe$; $Coll$
$\Rightarrow \{Fact1,\ \ell > \frac{R-r}{v_{max}} - T \geq \epsilon + \delta + T_{al}\}$
false $\vee$ $AllSafe$; $Coll$
$\Rightarrow \{Case\ b,\ PLogic\}$
false

The implication in the last case has been derived under the assumption

$$\ell > \frac{R-r}{v_{max}} - T \geq \epsilon + \delta + T_{al}$$

Hence the final general design decision is:

$$R - r \geq (T + T_{al} + \delta + \epsilon) \cdot v_{max}$$

This completes the verification of the correctness of the control design.

B Appendix B

Fact1: A potential collision is detectable if all aircraft are safe before a collision and if $T \leq T_{coll}$.

$$\Box(AllSafe\ ;\ \mathbf{true}\ ;\ Coll \Rightarrow \ell \geq T_{coll} - T)$$

Proof:

$$
\begin{array}{l}
AllSafe\ ;\ \mathbf{true}\ ;\ Coll \\
\Rightarrow \{Definitions\} \\
\forall\, id_1, id_2 : ID \bullet \\
(\lceil Pos(id_1) = p_{1b} \wedge Pos(id_2) = p_{2b} \rceil \\
\Rightarrow id_1 = id_2 \vee dist(p_{1b}, p_{2b}) \geq 2R\ ;\ \mathbf{true})\ ; \\
\exists\, id_3, id_4 : ID \bullet \\
(\lceil Pos(id_3) = p_{1e} \wedge Pos(id_4) = p_{2e} \rceil \\
\wedge\, id_3 \neq id_4 \wedge dist(p_{1e}, p_{2e}) < 2r) \\
\Rightarrow \{;\ \exists\} \\
\exists\, id_3, id_4 : ID \bullet \\
(\forall\, id_1, id_2 : ID \bullet \\
(\lceil Pos(id_1) = p_{1b} \wedge Pos(id_2) = p_{2b} \rceil \\
\Rightarrow id_1 = id_2 \vee dist(p_{1b}, p_{2b}) \geq 2R\ ;\ \mathbf{true}) \\
\ ;\ (\lceil Pos(id_3) = p_{1e} \wedge Pos(id_4) = p_{2e} \rceil \wedge \\
id_3 \neq id_4 \wedge dist(p_{1e}, p_{2e}) < 2r) \\
\Rightarrow \{;\ \forall\} \\
\exists\, id_3, id_4 : ID \bullet \forall\, id_1, id_2 : ID \bullet \\
(\lceil Pos(id_1) = p_{1b} \wedge Pos(id_2) = p_{2b} \rceil \\
\Rightarrow id_1 = id_2 \vee dist(p_{1b}, p_{2b}) \geq 2R\ ; \\
\mathbf{true})\ ;\ (\lceil Pos(id_3) = p_{1e} \wedge Pos(id_4) = p_{2e} \rceil \\
\wedge\, id_3 \neq id_4 \wedge dist(p_{1e}, p_{2e}) < 2r) \\
\Rightarrow \{df;\ \} \\
\exists\, id_3, id_4 : ID \bullet \forall\, id_1, id_2 : ID \bullet \\
((\lceil Pos(id_1) = p_{1b} \wedge Pos(id_2) = p_{2b} \rceil \\
\Rightarrow id_1 = id_2 \vee dist(p_{1b}, p_{2b}) \geq 2R)\ ; \\
\mathbf{true}\ ;\ \lceil Pos(id_3) = p_{1e} \wedge Pos(id_4) = p_{2e} \rceil) \\
\wedge\, (id_3 \neq id_4 \wedge dist(p_{1e}, p_{2e}) < 2r) \\
\Rightarrow \{ASM_0\} \\
\exists\, p_{1b}, p_{1e}, p_{2b}, p_{2e} : POSITION \bullet \\
dist(p_{1b}, p_{2b}) \geq 2R \wedge dist(p_{1e}, p_{2e}) < 2r \wedge \\
dist(p_{1b}, p_{1e}) \leq v_{max} \cdot (\ell + T) \wedge \\
dist(p_{2b}, p_{2e}) \leq v_{max} \cdot (\ell + T) \\
\Rightarrow \{Math\} \\
dist(p_{1b}, p_{2b}) - dist(p_{1e}, p_{2e}) \geq 2(R - r) \wedge \\
dist(p_{1b}, p_{1e}) + dist(p_{2b}, p_{2e}) \leq \\
\qquad 2 \cdot v_{max} \cdot (\ell + T) \qquad (1)
\end{array}
$$

Now we will use some properties of the function $dist$. The triangle inequality applied twice gives

$$
\begin{array}{l}
dist(p_{1b}, p_{1e}) + dist(p_{1e}, p_{2e}) + dist(p_{2e}, p_{2b}) \geq \\
\qquad dist(p_{1b}, p_{2b}) \\
\Rightarrow \{\mathsf{Symmetry}, \mathsf{Arithmetic}\} \\
dist(p_{1b}, p_{1e}) + dist(p_{2b}, p_{2e}) \geq \\
\qquad dist(p_{1b}, p_{2b}) - dist(p_{1e}, p_{2e}) \\
\Rightarrow \{(1)\} \\
2 \cdot v_{max} \cdot (\ell + T) \geq 2(R - r) \\
\Rightarrow \{\mathsf{Math}\} \\
\ell \geq \frac{R-r}{v_{max}} - T
\end{array}
$$

In order to have $\ell \geq 0$ we choose the sampling time such that $T \leq \frac{R-r}{v_{max}} = T_{coll}$.

Fact2 is proved in a similar way.