AIX-MARSEILLE UNIVERSITÉ
ECOLE DOCTORALE

Laboratoire d'Informatique et Systèmes / UMR 7020

Thèse présentée pour obtenir le grade universitaire de docteur

Discipline: MATHEMATIQUES ET INFORMATIQUE
Spécialité: Informatique

José Luis VILCHIS MEDINA

Modélisation des Systèmes Résilients en Logique Non-monotone. Application à UAV Solaire.

Modeling of Resilient Systems in Non-monotonic Logic. Application to Solar Power UAV.

Soutenue le 12 Décembre 2018 devant le jury composé de:

Andrei DONCESCU	LAAS, Toulouse	Directeur de thèse
Pierre SIEGEL	LIS, Marseille	Directeur de thèse
Jacques DEMONGEOT	Univ. J. Fourier, Grenoble	Rapporteur
Lakhdar SAIS	CRIL, Lens	Rapporteur
Amal EL FALLAH SEGHROUCHNI	LIP6, Paris	Examinatrice
Yves LACROIX	Université de Toulon	Examinateur
Vincent RISCH	LIS, Marseille	Examinateur

Acknowledgments

Tout d'abord, je tiens à remercier le Secrétariat à l'énergie (SENER), par l'intermédiaire du Conseil national mexicain pour la science et la technologie (CONACYT) [numéro de bourse 581317/412566], pour son soutien financier.

Je voudrais remercie aux rapporteurs, Jacques DEMONGEOT et Lakhdar SAIS, d'avoir accepter de lire cette thèse. Je remercie également tous les membres de jury d'avoir accepté d'assister à la presentation de ce travail.

Au Pr Sergio Gonzalez Rojo pour m'avoir contacté avec Pr Andrei Doncescu. Aux directeurs de thèse Pr Andrei Doncescu et Pierre Siegel pour m'avoir dirigé dans cette thèse. Aux directeurs, professeurs et membres du personnel administratif qui constituent le Laboratoire d'Informatique et Systèmes - LIS, en particulier à Sylvain Sené et à tous les membres du groupe CANA, pour m'avoir intégré dans l'équipe. Aux collègues de laboratoire, aux personnes qui ont contribué directement ou indirectement au cours de mon séjour de thèse.

Enfin, à mes parents, à mon frère et ma sœur pour avoir été une partie importante de ma motivation pendant tout le temps.

Firstly, I want to thank to the Secretariat of Energy (SENER) through the Mexican National Council for Science and Technology (CONACYT)[grant number 581317/412566] for their financial support provided.

I would like to thank the rapporteurs, Jacques DEMONGEOT and Lakhdar SAIS, for agreeing to read this thesis. I also thank all the jury members for agreeing to attend the presentation of this work.

To Pr. Sergio Gonzalez Rojo for having put me in contact with Pr. Andrei Doncescu. To my thesis directors to Pr. Andrei Doncescu and Pr. Pierre Siegel for having addressed me in this thesis. To the directors, professors and administrative staff that constitute the Laboratoire d'Informatique et Systèmes - LIS, especially to Sylvain Sené and to all those who make up the CANA group, for having included me in the group. To the laboratory colleagues, to the people who contributed directly or indirectly during my stay of my thesis.

Finally, to my parents, to my brother and sister for being a big part of my motivation during all the time.

En primer lugar, quiero agradecer a la Secretaría de Energía (SENER) a través del Consejo Nacional de Ciencia y Tecnología de México (CONACYT) [folio 581317/412566] por el apoyo financiero brindado.

Quisiera agradecer a los relatores, Jacques DEMONGEOT y Lakhdar SAIS, por aceptar leer esta tesis. También agradezco a todos los miembros del jurado por aceptar asistir a la presentación de este trabajo.

Al Pr. Sergio Gonzalez Rojo por haberme puesto en contacto con el Pr. Andrei Doncescu. A mis directores de tesis al Pr. Andrei Doncescu y al Pr. Pierre Siegel por haberme dirigido en esta tesis. A los directivos, profesores y personal administrativo que constituyen el Laboratoire d'Informatique et Systèmes - LIS, especialmente a Sylvain Sené y a todos los que integran el grupo CANA, por haberme incluido en el grupo. A los colegas de laboratorio, a las personas que contribuyeron directa o indirectamente durante mi estancia de mi tesis.

Por ultimo, a mis padres y hermanos por ser gran parte de mi motivación durante todo el tiempo.

Abstract

Cette thèse présente un modèle résilient pour piloter un avion basé sur une logique non monotone. Ce modèle est capable de gérer des solutions à partir d'informations incomplètes, contradictoires et des exceptions. C'est un problème très connu dans le domaine de l'Intelligence Artificielle (Artificial Intelligence (AI)), qui est étudié depuis plus de 40 ans. Pour ce faire, nous utilisons la logique des défauts pour formaliser la situation et trouver des actions possibles. Grâce à cette logique, nous pouvons transformer les règles de pilotage en défauts. Ensuite, lorsque nous calculons les solutions, plusieurs options peuvent en résulter. À ce stade, il existe un critère de décision opportuniste pour choisir la meilleure solution. Le contrôle du système se fait via la propriété de résilience. Nous redéfinissons cette propriété comme l'intégration de la logique non monotone dans le modèle de Minsky. En conséquence, il est démontré que le modèle de résilience proposé pourrait être généralisé aux systèmes intégrant une connaissance du monde contenant des situations, des objectifs et des actions. Enfin, nous présentons les résultats expérimentaux et la conclusion de la thèse en discutant des perspectives et des défis pour les orientations futures. Différentes applications dans d'autres domaines sont prises en compte pour l'intérêt du comportement du modèle.

Mots clés : Logique Non-monotone, Logique des défauts, Résilience, Intelligence Artificielle, Modèle de Marvin Minsky, Prise de Décisions, UAV, Représentation des Connaissances, Raisonnement Incertain, Systèmes Embarqués.

This thesis presents a resilient model to pilot an aircraft based on a non-monotonic logic. This model is capable of handling solutions from incomplete, contradictory information and exceptions. This is a very well known problem in the field of AI, which has been studied for more than 40 years. To do this, we use default logic to formalise the situation and find possible actions. Thanks to this logic we can transform the piloting rules to defaults. Then, when we calculate the solutions, several options could result. At this point an opportunistic decision criteria takes place to choose the better solution. The control of the system is done via the property of resilence, we redefine this property as the integration of the non-monotonic logic in the Minsky's model. As a result, it is shown that the proposed resilient model could be generalised to systems that incorporate a knowledge of the world that contains situations, objectives and actions. Finally, we present the experimental results and conclusion of the thesis discussing the

prospects and challenges that exist for future directions. Different applications in other fields are taken into account for the interest of the model's behavior.

Keywords: Default Logic, Resilience, Artificial Intelligence, Marvin Minsky's model, Decision-Making, UAV, Knowledge Representation, Reasoning under Uncertainty, Embedded Systems.

Esta tesis presenta un modelo resistente para pilotar un avión basado en una lógica no monótona. Este modelo es capaz de manejar soluciones a partir de informaciones incompletas, contradictorias y excepciones. Este es un problema muy conocido en campo de la Inteligencia Artificial (AI), que se ha estudiado durante más de 40 años. Para hacer esto, usamos la lógica por defectos para formalizar la situación y encontrar posibles acciones. Gracias a esta lógica, podemos transformar las reglas de pilotaje en defectos. Luego, cuando calculamos las soluciones, pueden surgir varias opciones. En este punto, tiene lugar un criterio de decisión oportunista para elegir la mejor solución. El control del sistema se realiza a través de la propiedad de resilencia. Redefinimos esta propiedad como la integración de la lógica no monotónica en el modelo de Minsky. Como resultado, se muestra que el modelo resistente propuesto podría generalizarse a sistemas que incorporan un conocimiento del mundo que contiene situaciones, objetivos y acciones. Finalmente, presentamos los resultados experimentales y la conclusión de la tesis sobre las perspectivas y los desafíos que existen para futuras direcciones. Se tienen en cuenta diferentes aplicaciones en otros campos para el interés del comportamiento del modelo.

Palabras claves: Lógica No monótona, Lógica por Defectos, Resiliencia, Inteligencia Artificial, Modelo de Marvin Minsky, Toma de Desiciones, UAV, Representación del Conocimiento, Razonamiento bajo Incertidumbre, Sistemas Embebidos.

Contents

List of Figures

List of Tables

Introduction

This thesis presents a resilient model to pilot an Unmanned Aerial Vehicle (UAV) based on a non-monotonic logic. This model is capable of handling solutions from incomplete and contradictory information. This is a very well known problem in AI, which has been studied for more than 40 years. An UAV is any plane without a pilot and generally the control system is on the ground. It should be considered that the plane does have a pilot that is in a fixed station on land. However, the problem does not change since the UAV must have control of the navigation, trajectory, legislations to respect... There are UAV that use control methods for autonomous purposes. Nevertheless, they are based on petri nets or differential equations, for discrete and continuous modeling, respectively. In both cases, they can not solve the problem of incomplete and contradictory information. For this purpose, we are going to model the behavior of a pilot. On one hand, he could have information from the control tower in contradiction with the current situation, it could be an emergency. On the other hand instruments on board could show partial information. This leads us to also consider the weather conditions because climate changes constantly. It is with all this that the pilot must make decisions to carry out his mission. To do this, we use default logic to formalise the situation and find possible conclusions. At this point an opportunistic decision criteria takes place to choose the better solution, taking into account security, priority... The control of the system is done via the property of resilence. We redefine this property as the integration of the non-monotonic logic in the Minsky's model. Thanks to this, we can determine a horizon of convergence. This horizon can be short or long according to the current situation and goal. In the same way, this model will allow us to know the behavior. As a result, it is shown that the proposed resilient model could be generalised to systems that incorporate a knowledge of the world that contains situations, objectives and actions. Finally, the implementation in a microcomputer is performed. The composition of the thesis is described below.

Chapter 1 introduces the external and internal parts of an airplane, as well as the theory of flight where the lift, thrust, drag and gravity interact. Similarly, we explain the phases of a flight and the challenges of piloting. For this it is essential to study the behavior of a pilot. Because in reality, a pilot must take into account not only the information he sees in the cabin (air speed, altitude, variometer, compass...) but also what happens outside the cabin. Throughout the mission, a pilot must be in communication with the control tower following the orders, as there is an airspace regulation that must be respected. Pilots that

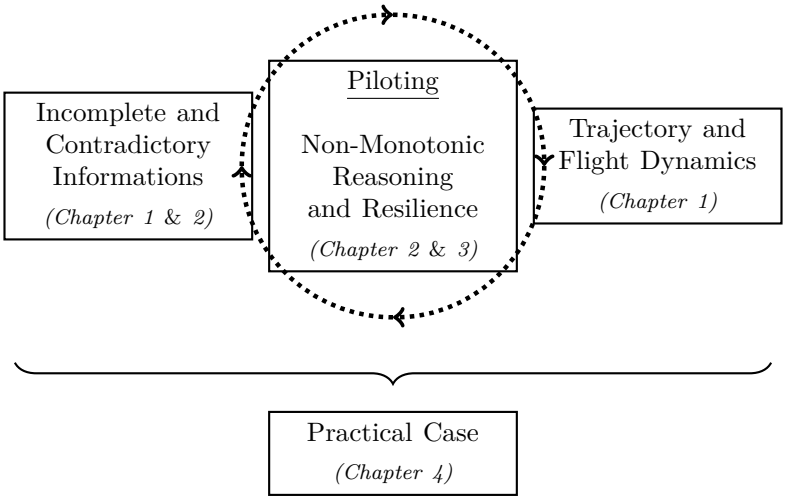


Figure 0.1: Composition of the different chapters of the thesis.

fly, only have a fixed radio frequency to communicate with the control tower. This makes all the pilots listen to the communication at the same time. Even if the control tower communicates with a particular pilot. In addition, if the ground controller sends orders to a specific pilot but another or others have an emergency, they must violate the airspace regulations to solve their problems. On the other hand, the climate changes constantly (wind, snow, hail...) and can affect the sensors. The airspeed sensor provides an important information that allows an airplane to lift. If it does not have the necessary airspeed, the airplane will not generate the necessary force to descend in altitude. These factors can easily lead to contradictions, changing to a reasoning under uncertainty, despite this, a pilot must make decisions to carry out his mission.

Chapter 2 presents the formalisation of the problem, which allows reasoning under uncertainty in the presence of incomplete and contradictory information. We will see that classical logic has limitations and we must move on to non-monotonic reasoning. For this we used a particular logic to solve the problem, default logic. Thanks to this logic we can transform the piloting rules to default rules. Then, when we calculate the solutions several options could result. Once the solutions have been calculated, an opportunistic notion is used to choose the best option according to criteria such as security, energy, priority... of each rule listed. The implementation in SWI-Prolog calculates the solutions for a given situation.

Chapter 3 concerns the control part, but also the definition and properties of resilience. The KOSA model is described containing the set of knowledge and the sub-set of objectives, situations and actions. The fact of being able to formalise the knowledge in this manner allows us to incorporate it into the Minsky's model. To make a better capture of the property of resilience in our model. For that, new definitions such as short-term and long-term objectives had been de-

scribed. Consequently, the discrete and continuous behavior of the KOSA model is presented.

Chapter 4 presents the experimental results of this thesis. A resilient decision making system for an UAV based on non-monotonic logic is embedded on a microcontroler. We explain the configuration of SWI-Prolog for the microcomputer and to interface it with sensors and others electrical circuits.

Finally, the conclusion of the thesis discussing the prospects and challenges that exist for future directions that still require study. Different applications in other fields are taken into account for the interest of the model's behavior under other complex dynamic environments.

1 Challenges of Piloting and Motivation

The objective of this section is to present the external and internal parts of an airplane, as well as the theory of flight where the force of sustentation, thrust, drag and gravity interact. Similarly, Bernoulli's principle that is the fundamental principle for the flight of aircraft. Equally important, to present the phases of a flight. Subsequently, the challenges of piloting and finally the motivation of this research topic are exposed.

Summary

1.1. Background and Context

We are going to study the rules of piloting and how a pilot manages these rules in the face of unusual events. Through the history of aviation, piloting have had different problems that are related to decision-making with incomplete and contradictory information. Pilots should consider as much as the variables of the airplane, communication with the control tower, as well as the climatological conditions. For the first two, there is a set of instruments inside the cabin. It shows information such as: airspeed, vertical speed, turning angles... In addition, the weather conditions change constantly. Airplanes have instruments to know about it. Radars and navigation systems that give to pilots the climate update. But the storms, snow, wind and fog are unpredictable. All this does that a pilot could make bad decisions.

An answer to this problem was the automatic flight control. Automatic flight control dates back to the 1920s, and during the II World War there were already rudimentary autopilots. The idea was that automation could mitigate pilots' flight routines and allow them to focus on situational awareness and other care

priorities. Technology evolved and by the 20th century, autopilots were more sophisticated. Especially in safety, detecting problem quickly and effectively in comparison with human pilots. A person sometimes may not see the increase of a variable in an indicator, instead a computer can detect signs of danger and send an alert.

Hereafter, we will briefly present the most relevant information about the different parts of the aircraft, theory of flight, phases of flight and trajectory analysis. Finally, challenges of piloting and motivation will be present.

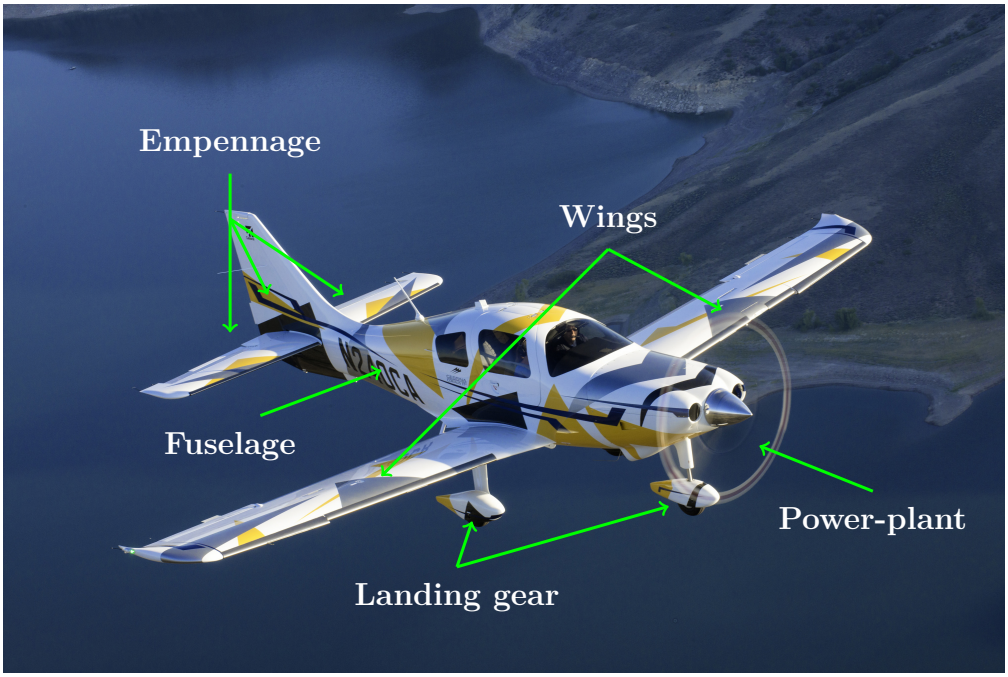
We all know that an airplane has two wings, thanks to them will create the necessary force to fly. The distance that go from one wing to the other is called wingspan, Figure 1.1. The stiffness of a wing will depend mainly on the mast and internal structure. The coating can be made of wood, metal or plastic. There are free spaces within the structure that are generally used for the fuel. It is known as leading edge and trailing edge to the front and back sides of the wing, respectively. The trailing edge is articulated, this is to orient the surface. These orientations modify the curving of the wing allowing flying at low speed.

At the end of the wings are the ailerons. When the airplane is flying these movements allow an inclination to the left or to the right. These have an opposite movement, up or down. The fuselage is the structure that joins the wings ensuring the rigidity of the airplane. Like the wings, the fuselage has a coating that can be made of wood, metal or plastic. The positioning of the wings with the fuselage can be of two levels: wings down or wings up. At the rear of the fuselage are the stabilizers. They usually have a cross shape. Where the vertical part corresponds to a fixed part, called drift, and further back has a mobile surface which is articulated for the steering rudder. The horizontal part is constituted by a fixed part in which the depth rudder is articulated, Figure 1.1 (Nancy 2016). The aim of the landing gear is to ensure the airplane's handling and to cushion the effects suffered when the airplane touches the ground.

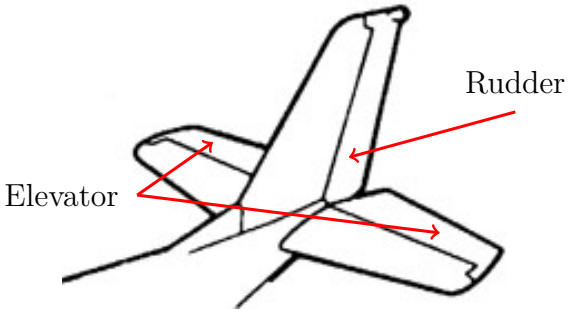
In the cabin we have the cockpit, to get inside it is necessary to climb over the wing, this is for low wing aircrafts. The pilot usually occupies the left seat. In this way he has access to all the controls. The steering wheel and rudder constitute the main flight controls. Sometimes the steering wheel is replaced by a handle, however it has the same function, Figure 1.2. The steering wheel can move forward and backward allowing to control the depth. In the same way, the steering wheel is used laterally from left to right which control the ailerons. And the pedals control the steering rudder, but this it used principally to control the airplane on the ground, Figure 1.3.

Instruments on Board

There are many instruments in the cabin, as we can see in the Figure 1.2. However, only six are essential to be able to fly. Commercial, private airplanes and gliders, they have it (DGAC/SFACT 1992). Generally, this set of indicators



(a) Parts of an airplane



(b) Stabilizers

Figure 1.1: Parts of an airplane.

are:

- **Altimeter**: shows the aircraft's altitude (in feet) above sea-level. As the aircraft ascends, the altimeter to indicate a higher altitude and vice versa.
- **Airspeed indicator**: shows the aircraft's speed (in knots) relative to the surrounding air. It works by measuring the ram-air pressure in the aircraft's Pitot tube relative to the ambient static pressure.
- **Vertical speed indicator**: sometimes called a variometer or also rate of climb indicator, senses changing air pressure, and displays that information to the pilot as a rate of climb or descent in feet per minute or meters per second.

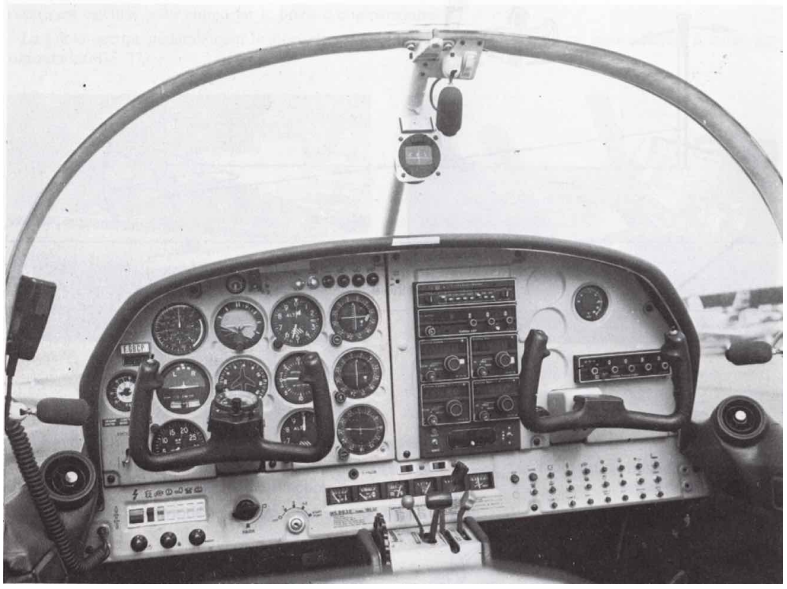


Figure 1.2: Cabin of an airplane.

- **Attitude indicator**: also known as an *artificial horizon*. Shows the aircraft's relation to the horizon. From this a pilot can tell whether the wings are level *roll*, if the aircraft nose is pointing above or below the horizon *pitch*, Figure 1.6.
- **Turn indicator**: This instrument includes the Turn-and-Slip Indicator and the Turn Coordinator, which indicate rotation about the longitudinal axis. When an airplane begins to turn a centrifugal force will be generated and will make altitude lose, the challenge of the pilot is to compensate for the fall in altitude by increasing the airspeed and also increasing the angle of attack (positive pitch). In other words, it is important to keep the indicator in the center so that you have a planar turn, or else it will spill, that is, the airplane will rotate orthodoxy.
- **Heading indicator**: displays the aircraft's heading with respect to magnetic north when set with a compass.

Theory of Flight

Flying is a phenomenon that has long been part of the nature. Birds fly not only with the beating of their wings but also by planning long distances when they spread them. Thanks to the principles of physics it is possible to fly. Based on these principles, man has created airplanes and has been able to counteract the force of gravity. A flight heavier than air is possible thanks to the balance of four physical forces: **lift**, **drag**, **weight** and **thrust**. For this, the plane must have

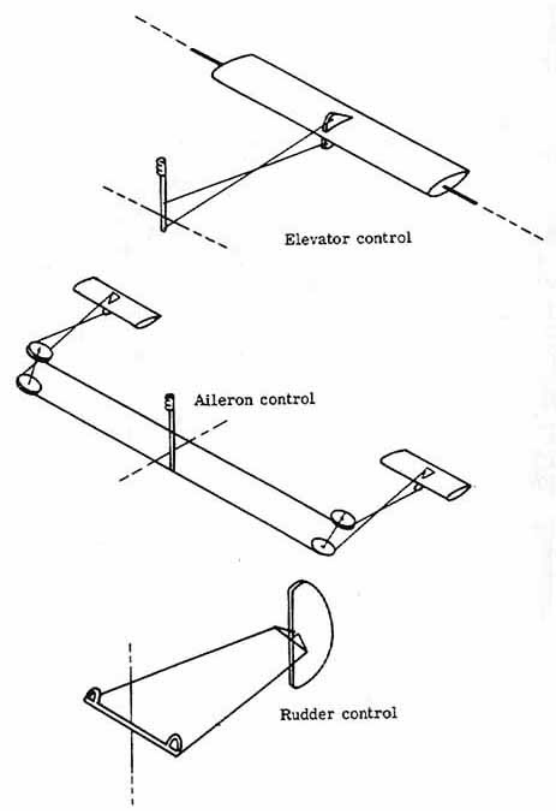


Figure 1.3: Controls of an airplane.

a balance of its lift with its weight, and also its thrust must exceed its drag. An airplane uses its wings to generate lift and its engines for the thrust, Figure 1.5. To reduce the drag the aircraft has a smooth shape and the weight will be defined by the material with which it is made. To stay in the air, the forces acting on the airplane must be null: $\overrightarrow{L} + \overrightarrow{D} + \overrightarrow{T} + \overrightarrow{W} = \overrightarrow{0}$. The air is a means used by the airplanes to generate lift. It is composed of particles of 78 % nitrogen, 21 % oxygen and 1 % other gases and it has physical properties such as expansion, contraction, fluidity, mass and density. It should be mentioned that the density of the air (ρ) depends of the height (h), the equation that defines it is given by:

$$\rho = \rho_{sealevel} \cdot \exp^{-\beta \cdot h} = 1.255 Kg/m^3, \beta = \frac{1}{9.042m} \tag{1.1}$$

And for dynamic pressure ($\tilde{q}$), it is given by:

$$\begin{aligned} V_{air} &= [v_x^2 + v_y^2 + v_z^2]_{air}^{1/2} \\ \tilde{q} &= \frac{1}{2} \cdot \rho \cdot h \cdot V_{air}^2 \end{aligned} \tag{1.2}$$

Where V_{air} is the airspeed in the three components $v_{x,y,z}$.

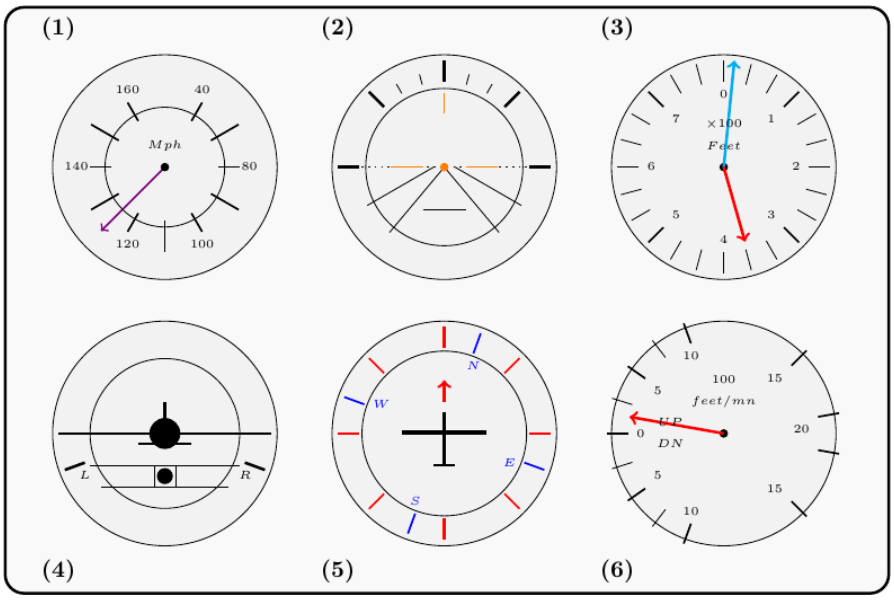
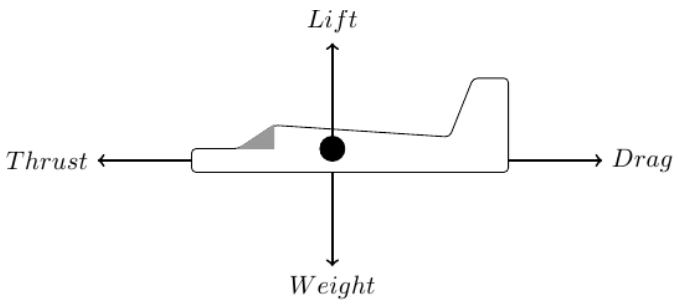
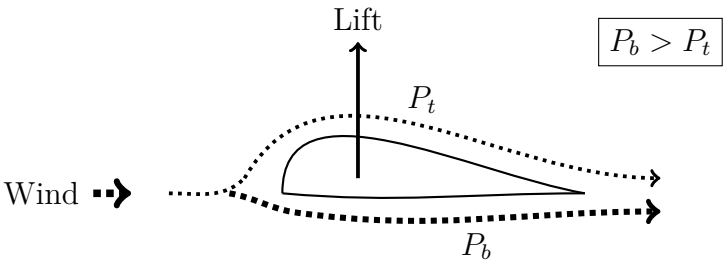


Figure 1.4: Cockpit: six basic instruments.



(a) Forces on an airplane



(b) Bernoulli's principle

Figure 1.5: Aerodynamic principles of an airplane.

The fundamental principle that makes airplanes fly is called lift. This aerodynamic effect was described by Bernoulli studying the liquids. However, air can be considered as a fluid with certain properties. This effect happens when an object with a determinate shape and surface crosses the air. It is then that

the particles of the air will have two trajectories. Through the upper part of the wing, the particles of the air will travel faster generating an out of low pressure. At the bottom of the wing the air particles will travel at a lower speed generating a higher pressure force. The ideal way to have the maximum strength of lift is shown in the Figure 1.5, a classic form of an airplane wing. Depending on the speed of the body, in this case of the airplane, there will be more or less lift. If there is more speed then it will have more lift and vice versa. The angle of attack is the angle formed between the wing and the direction of the air. This is an important condition which allows to increase or decrease the lifting force.

Lift

For an airplane starts flying, it must have a force that is equal or greater than the weight (mass $\times$ force of gravity). This force is called lift and it is created by the flow of air on a supporting surface. The shape of the lifting surface causes a flow of air with higher velocity in the upper part than in the lower part. The increase in speed causes a low pressure. However, the opposite occurs in the lower part. The air pressure is greater under the lifting surface, this generates a lift force. To better understand this phenomenon, it is necessary to make use of two important equations in physics. The pressure variations of the air flow are described by the Bernoulli's equation. This equation was discovered by the Swiss mathematician Daniel Bernoulli to explain the pressure variations exerted by water currents. The Bernoulli's equation is described as:

$$P + \frac{1}{2} \cdot \rho \cdot V^2 = constant \tag{1.3}$$

Where P is the pressure, ρ is the density of the fluid and V is the velocity of the moving object or fluid. The second important equation that must be taken into account is the continuity equation. Which simply says that given a flow, the product of density, cross-sectional area and velocity is constant. This equation is defined as:

$$\rho \cdot A \cdot V = constant \tag{1.4}$$

Where ρ is the density, A is the area transversal and V is the velocity. Making use of the Equation (1.3) and (1.4) it can be better understood how the air flows on the surface and generates the lift force. The shape of a bearing surface is usually asymmetric, because the area of the upper part is larger than the lower one. When a flow of air passes through the supporting surface, it will move more through the upper part than through the lower one. According to the continuity equation, having a greater displacement will cause an increase in velocity to occur. The Bernoulli equation shows that an increase in velocity has an effect on the decrease in pressure. The faster the flow, the lower the pressure. The air flowing on a bearing surface will have a decrease in pressure in the upper part

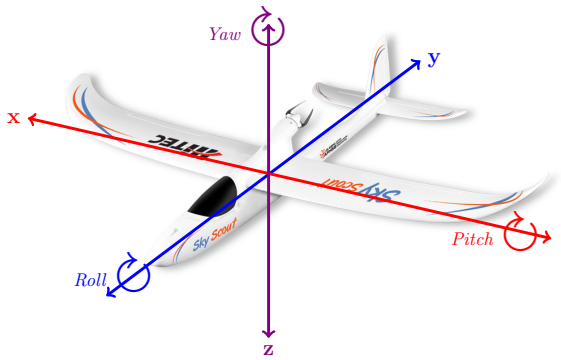


Figure 1.6: Airplane body axes.

and an increase in the lower part. Resulting a force of total pressure upwards. This pressure force is lift. The design of the shape of the lifting surface depends on the usefulness of the plane. For this reason there is no predetermined form. Engineers use the lift coefficient to know a lift measurement obtained for a particular shape. Lift is directly proportional to the dynamic pressure and wing area.

$$lift = C_L \cdot (\frac{1}{2} \cdot \rho \cdot V^2) \cdot S \tag{1.5}$$

Where S is the wing area and dynamic pressure is $\frac{1}{2} \cdot \rho \cdot V^2$. In the design of a wing of an airplane it is very important to have a high lift coefficient.

In Figure 1.6, ***pitch*** is the angle formed by the airplane when has rotated around "x-axis". Similarly, ***roll*** is formed by the airplane when has rotated, but around "y-axis", and ***yaw*** is the angle formed when the airplane rotates around "z-axis".

When the pilot turns to the left or right forces such as as centrifugal and tangential will appear. These two forces influence the trajectory of an airplane. The pilot must compensate for these effects of these two forces to follow the desired trajectory. When you turn left or right, due to aerodynamic effects due to these two forces, the plane will lose altitude. Then the pilot must increase the angle of attack and increase the speed to compensate for that loss of altitude, Figure 1.7.

Drag

Any object that crosses the air will have a resistance to the flow of it. This resistance is called drag. Drag is the product of many physical phenomena. The wings of a plane are designed to be as smooth as possible to reduce drag force to the maximum. As a lift, drag is proportional to the dynamic pressure and the area over which it acts. The coefficient drag, analogously that the lift coefficient. This is a measure of the amount of dynamic pressure transformed into drag. In

(a) straight path (b) turing

Figure 1.7: Forces acting in an airplane.

practical terms, engineers usually design wings with a drag coefficient as low as possible in magnitude. Low drag coefficient are more interesting because the efficiency of the plane increased when drag decreases.

$$drag = C_D \cdot (\frac{1}{2} \cdot \rho \cdot V^2) \cdot A \tag{1.6}$$

Weight

Weight is a limiting factor when airplanes are deferred. The higher the weight of the plane requires more lift compared to a lighter plane. A higher weight also requires more thrust to accelerate on land. But also in a light aircraft the distribution of the weight is important. It must be balanced for the flight, much weight on the back or front can cause instability. The weight can be calculated thanks to Newton's second law:

$$W = m \cdot g \tag{1.7}$$

Where W is weight, m is mass and g is the acceleration due to gravity on Earth.

Thrust

When an object is propelled it generates different physical principles. Which are thermodynamic, aerodynamic, fluid and physical mechanics. Thrust is a force that is described by Newton's second law. The basic form of this law is defined as:

$$F = m \cdot a \tag{1.8}$$

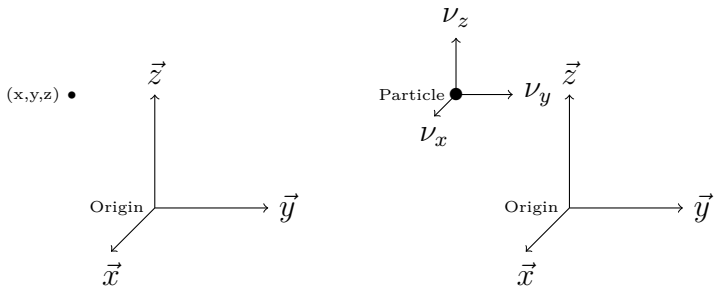
Where F is force, m is mass and a is acceleration. Acceleration is the change of speed with respect to time. Therefore, Thrust is produced by the acceleration of a mass of air.

Flight Dynamics

A manner of studying aircraft dynamics is to use Newton's laws. The equations of motion of a particle can be studied from a Newtonian frame. The translation movement causes an object to change its position with respect to a frame of reference. The frame of reference is fixed and the position of the particle is $r = (x, y, z)$, Figure 1.8. The speed of the particle is defined as below:

$$v = \frac{\partial x}{\partial t} = \dot{x} = [\dot{x}, \dot{y}, \dot{z}] = [\nu_x, \nu_y, \nu_z] \tag{1.9}$$

The movements of an airplane has rotations and translations in the three axes.



(a) Position (b) Speed

Figure 1.8: Newtonian frame.

When it is tilting up or down, turning to the right or to the left, etc. this will add other forces (DGAC/SFACT 1992). These forces that act on in any direction, can be described by Newton's laws. Newton's first law says that if there is no force (F) on an object with a mass (m), it will remain at rest or will continue in rectilinear motion at a constant speed (v). Retaining its momentum: $\frac{\partial(m \cdot v)}{\partial t} = 0$, $m \cdot v(t_1) = m \cdot v(t_2)$. The second law says that if a force acting on a fixed mass object will have a speed change with acceleration proportional to the direction of that force. $F = m \cdot a = \frac{\partial(m \cdot v)}{\partial t}$. For three dimensions it should consider force $F = [f_x, f_y, f_z]$, $\mathbb{I}_3$ is an unitary matrix, so acceleration will be defined as:

$$a = \frac{\partial v}{\partial t} = \frac{1}{m} \cdot F = \frac{1}{m} \cdot \mathbb{I}_3 \cdot F = \begin{bmatrix} 1/m & 0 & 0 \\ 0 & 1/m & 0 \\ 0 & 0 & 1/m \end{bmatrix} \cdot \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{1.10}$$

We know that acceleration is defined as the rate of change in speed:

$$\dot{v} = \frac{\partial v}{\partial t} = \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix}_l = \frac{1}{m} \cdot F = \begin{bmatrix} 1/m & 0 & 0 \\ 0 & 1/m & 0 \\ 0 & 0 & 1/m \end{bmatrix} \cdot \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{1.11}$$

And speed is defined as the rate of change in distance:

$$\dot{r} = \frac{\partial r}{\partial t} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_l = v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_l \tag{1.12}$$

Eventually, the third law says that to every action there is a contrary force of equal magnitude (Hull 2007)). Remembering the balance of aerodynamic forces, we know that lift (F_l) can be calculated as following:

$$F_l = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}_l = [F_{gravity} + F_{aerodynamics} + F_{thrust}]_l \tag{1.13}$$

$$\dot{v}(t) = \frac{\partial v(t)}{\partial t} = \frac{1}{m} \cdot F = \begin{bmatrix} f_x/m \\ f_y/m \\ f_z/m \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_l \tag{1.14}$$

The speed of the airplane is a variable that must be considered all the time t, as previously mentioned this it is proportional to lift. To calculate it, we should solve the next equation:

$$v(T) = \int_0^T \frac{\partial v(t)}{\partial t} + v(0) = \int_0^T \frac{1}{m} \cdot F \partial t + v(0) = \int_0^T a \cdot \partial t + v(0) \tag{1.15}$$

Where $v(0)$ is the initial speed in three dimensions. This is the solution in space:

$$\begin{bmatrix} v_x(T) \\ v_y(T) \\ v_z(T) \end{bmatrix} = \int_0^T \begin{bmatrix} f_x/m \\ f_y/m \\ f_z/m \end{bmatrix} \partial t + \begin{bmatrix} v_x(0) \\ v_y(0) \\ v_z(0) \end{bmatrix} = \int_0^T \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \partial t + \begin{bmatrix} v_x(0) \\ v_y(0) \\ v_z(0) \end{bmatrix} \tag{1.16}$$

To know the distance traveled, where $r(0)$ is the reference in three dimensions:

$$\dot{r}(t) = \frac{\partial r(t)}{\partial t} = v(t) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{bmatrix} \tag{1.17}$$

$$r(T) = \int_0^T \frac{\partial r(t)}{\partial t} \partial t + r(0) = \int_0^T v(t) \partial t + r(0) \tag{1.18}$$

$$\begin{bmatrix} x(T) \\ y(T) \\ z(T) \end{bmatrix} = \int_0^T \begin{bmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{bmatrix} \partial t + \begin{bmatrix} x(0) \\ y(0) \\ z(0) \end{bmatrix} \tag{1.19}$$

The gravitational force ($m \cdot g$) is independent of the position. The gravitational acceleration is g. Where $g_0 \approx 9.807\ m/s^2$ at the earth's surface:

$$(F_{gravity})_l = (F_{gravity})_E = m \cdot g_E = m \cdot \begin{bmatrix} 0 \\ 0 \\ g_o \end{bmatrix}_E \tag{1.20}$$

To know the dynamics of the trajectory with no aerodynamics force and constant gravity, initial conditions for speed and position are: $v_x(0) = v_{x_0}$, $v_z(0) = v_{z_0}$, $x(0) = x_0$ and $z(0) = z_0$. The components of acceleration and speed are defined as:

$$\begin{aligned} \dot{v_x}(t) &= 0 \\ \dot{v_z}(t) &= -g(+z_{up}) \\ \dot{x}(t) &= v_x(t) \\ \dot{z}(t) &= v_z(t) \end{aligned} \tag{1.21}$$

Finally, we should solve these equations:

$$\begin{aligned} v_x(T) &= v_{x_0} \\ v_z(T) &= v_{z_0} - \int_0^T g \partial t = v_{z_0} - g \cdot T \\ x(T) &= x_0 + v_{x_0} \cdot T \\ z(T) &= z_0 + v_{z_0} \cdot T - \int_0^T g \cdot t \partial t = z_0 + v_{z_0} \cdot T - \frac{g \cdot T^2}{2} \end{aligned} \tag{1.22}$$

The dynamic analysis of an airplane is very well known with the mathematical model that involves accelerations and speeds in the three axes, but there are many restrictions in order to have linear equations(Duke, Antoniewicz, and Krambeer 1988) and find a solution within a reasonable period of time.

Next, we present a complete model with respect to the previous equations which do not contain parameters such as the angle of the runway, aerodynamic slope, etc. This model takes into account the movements in the wheels and aircraft pitch caused by the deceleration on the runway. For this, the angle of the runway is evaluated, considered as a constant in some parts of the runway, we should considerate two new variables as α (incidence) and q (pitching speed).

Applying the principles of the dynamics of solids, we obtain the mechanical equations of soil and flight. Obtaining the following system of equations, for the two axes. Where M_v is the mass of aircraft, V is the ground speed, β is the skid, V_{air} is the air speed, C is the aerodynamic coefficient. $F_{y,av,ar}$ is the front or back tire drift force, g is the gravity, ϕ is the roll angle, J is the moments of inertia, p is the roll speed. d_f is the front wheelbase, d_r is the back wheelbase, d_t is the stub axle, K is the rigidity of the shock absorber, R is the amortization, Y is the center of gravity with ground reference. F is the thrust, γ is the runway slope, α is the angle of incidence, N is the static charge (Villaumé 2002).

For lateral axe:

$$
\begin{aligned}
M_v \cdot V(\dot{\beta} + r) &= \frac{\rho}{2} \cdot V_{air}^2 \cdot S \cdot C_y + F_{yav} + F_{yar,1} + F_{yar,2} \cdot M_v \cdot g \cdot \phi \\
J_{zz} \cdot \dot{r} - J_{xz} \cdot \dot{p} &= \frac{\rho}{2} \cdot V_{air}^2 \cdot S \cdot l \cdot C_n + d_f \cdot F_{yav} - d_r \cdot (F_{yar,1} + F_{yar,2}) + d_{t,ar} \cdot (F_{xar,2} - F_{xar,1}) \\
-J_{xz} \cdot \dot{r} + J_{xx} \cdot \dot{p} &= \frac{\rho}{2} \cdot V_{air}^2 \cdot S \cdot l \cdot C_l - K_\phi \cdot \phi - R_\phi \cdot p \\
\dot{\phi} &= p \\
\dot{\psi} &= r \\
\dot{Y} &= V \cdot (\psi + \beta)
\end{aligned}
\tag{1.23}
$$

For longitudinal axe:

$$
\begin{aligned}
M_v \cdot \dot{V} &= -\frac{\rho}{2} \cdot V_{air}^2 \cdot S \cdot C_x + F - F_{xar,1} - F_{xar,2} - M_v \cdot g \cdot \gamma_{runway} \\
J_{yy} \cdot \dot{q} &= \frac{\rho}{2} \cdot V_{air}^2 \cdot S \cdot l \cdot C_m - K_\alpha \cdot \alpha - R_\alpha \cdot q + d_f \cdot (N_{ar,1} + N_{ar,2}) - h \cdot (F_{xar,1} + F_{xar,2}) \\
\dot{\alpha} &= q \\
\dot{X} &= V
\end{aligned}
\tag{1.24}
$$

Actually, computers on board have programmed these kind of equations above presented. Thanks to these equations solutions such as pitch, roll and yaw angles are calculated. The main purpose is to know the angles and follow a desired path previously programmed, knowing informations such as lineal speed and rotational accelerations, to control the airplane by engines. To understand how a pilot does a flight, it is important to mention that a flight is segmented in different states. These are described in the next section.

1.1.1. Phases of Flight

We begin by describing the basic circuit called traffic pattern. It is one of the basic maneuvers to take-off and land. However, it contains the necessary rules to carry out, even a long flight. Next, we explain the different states of flight for an airplane. Since most of these states are the same for motor-gliders. First of all, the pilot needs to know airplane states, so he uses the cockpit. The cockpit is a set of instruments on board that displays parameters such as ***airspeed*** (Miles/h),

artificial horizon (pitch and roll), ***variometer*** (Feet/s), ***altitude*** (Feet), ***compass***,... Traffic pattern circuit, Figure 1.9, has different flight stages. It starts at the point S_p where the airplane is in *Rest*. When the pilot is ready and he has the authorization, he increases all the engine power to get a right airspeed to take-off (point a). This is airplane should climb to a suitable height (point b). After that, the pilot should turn the yoke to the left making an orthogonal path to the runway (point c). At this point, he turns again the yoke to the right having constant airspeed, constant altitude and zero vertical speed. When he arrives to the point d he will prepare to land. Turning the yoke to the right, decreasing in altitude and having negative vertical speed, until arrive to the point e. Once again, he should turn the yoke to the right to continue decreasing in altitude and having a stable roll and negative pitch, until point f. After this point, airplane touches the ground. Final point F_p is where airplane state is again in *Rest*.

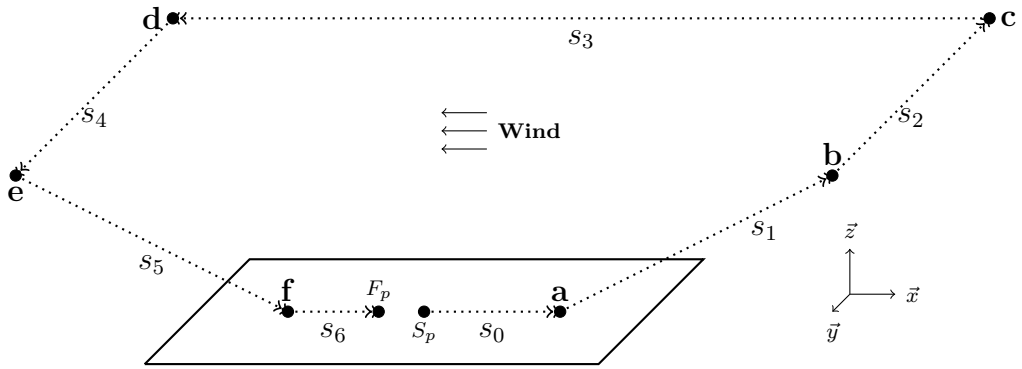


Figure 1.9: Traffic Pattern: basic circuit.

1.1.2. Trajectory Analysis

The segmentation of a flight makes each stage more understandable, it can be analyzed in the following way. This assignment of stages was made by us, however, we take them of the flight manual, having eight states but the most relevant states are five, Figure 1.10. To travel from point A to point B, an airplane has to take off, climb, descend and land, Figure 1.9. The traffic pattern circuit could be represented as follows:

- **Rest (RS):** When an airplane is at rest, airspeed and variometer indicators are approximately zero and the engine may be off. This could be happen before take-off or after landing.
- **Start (ST):** When a pilot has authorization to take-off, this is the initialization to have a necessary airspeed before take-off. Here the airplane is going to take a force greater than its weight. To take-off a lot of energy must be spent to reach the right speed and increase the angle of attack to

have more lift and increase in altitude. This is necessary to overcome the weight of the airplane. To maintain a constant slope climbing, a decreasing airspeed needs a pitch up variation. In other words, a pilot must increase the angle of attack to climb with a constant slope, if airspeed is not enough.

- **Takeoff (TO):** Once an airplane exceeds the speed limit to take-off. A pilot must pull the yoke to have a positive pitch ($0° < \measuredangle < 60°$), this will cause that the airplane increases its altitude and variometer will be positive.
- **Climb (CM):** After take-off, a pilot will keep the angle of attack constant ($0° <$ pitch $\measuredangle < 60°$), increasing in altitude.
- **Bank Turn (BT):** When a pilot wants to turn to the right or to the left, this will allow him change the path direction. Turning to turn the right, a positive roll is $0 <$ roll $\measuredangle < 60$, to turn to the left, a negative roll is $-60° <$ roll $\measuredangle < 0°$.
- **Steady flight(SF):** A flight in a straight line is considered such as stable state, however, this it will happen when an airplane has a desired altitude. So pitch and roll angles should be $\approx 0°$. In this stage the balance of the 4 forces must be fulfilled. In most cases, computer on-board will be adjusting the variations to maintain this balance.
- **Descend (DC):** When a pilot wishes to descend in altitude, either to converge at a desired altitude, evade an obstacle or to start landing stage. A negative pitch must be done, $-60° < \measuredangle < 0°$. In the same way such as climbing stage, if an airplane has not much airspeed then the pilot should increase the angle of attack, if not airplane will fall down.
- **Final approach(FA):** This is a state which a pilot decided to land, we say a pre-landing. Such as take-off stage, a pilot needs an authorization from the control tower. If a pilot has not a permission then he can not land, but if he has an emergency then he can land, so he will break the principle of the legislation.
- **Landing(LD):** It is the final stage of a flight, when an airplane touchdown. Its altitude decreases and variometer indicates $\approx 0°$ or a negative vertical speed. It is the most complicated part of a flight because a pilot should find the right airspeed and angle of attack to touchdown at runway properly.

In view of the above and having regard that a mission has a start point and final point. Next, we will see how a pilot plans a mission.

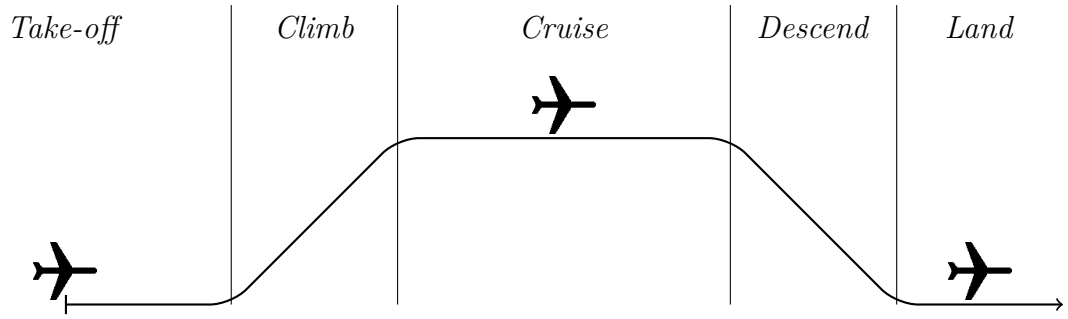
1.2. Challenges of Piloting

The principle of navigation of an airplane is made through three basic elements: a chronometer, airspeed and distance. Before flying, a pilot makes a preliminary plan on a map of the points he wishes to go, marking the starting point and the point of arrival. Since the maps show the distances between cities, mountains, even lakes, a pilot must do a basic calculation to know the time it will take to get to a point of interest. Knowing that $v = \frac{d}{t}$ or $t = \frac{d}{v}$. Where d is the distance from one point to another on the map, v is the airspeed of the airplane, and t is the time it will take to reach to final point. This technique is valid as long as the weather conditions are ideal, this means, when there is no wind. However, when this does not happen, there is a drift in the same wind direction. This directly affects the flight path, because the original plan would no longer be followed. For this a pilot must correct this drift, Figure 1.10 (Nancy 2016). The simplest path is a straight line, but there is no certainty that it is an environment without obstacles or without disturbances of the climate (wind, rain, etc.), so a pilot should be able to compensate and control these uncertain effects. A pilot must not only be hanging on the information he sees in the cabin and what happens outside during a flight. Throughout the course he must also be in communication with the control tower. Knowing that there is only one fixed radio frequency to communicate with all the other pilots that are flying as well. Also taking into account that aeronautical regulations must be respected. Believe it or not, this demands a lot of concentration and can cause bad decisions. On one hand, it could have the information of the control tower in contradiction with the current situation, e.g. an emergency. On the other hand the instruments could show partial informations, Figure 1.5. This leads us to also consider the weather conditions. Because the climate changes constantly. With all this a pilot must make decisions to carry out his mission, Figure 1.10.

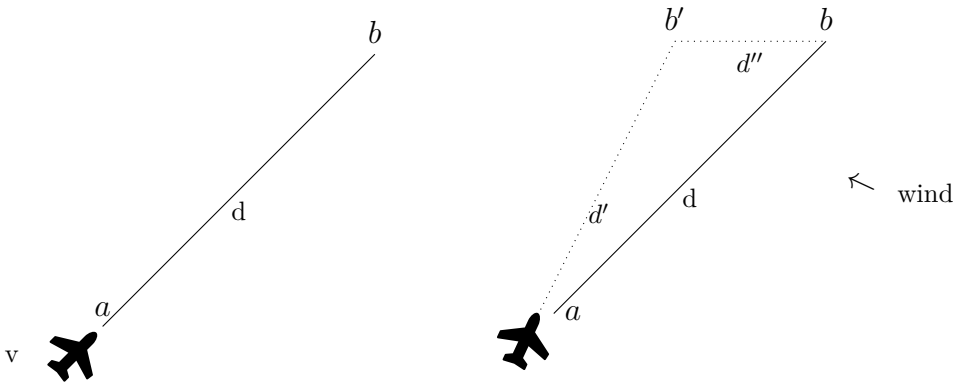
1.3. Motivation

In this thesis we have a dual purpose. In the first place, to develop a model based on Non-Monotonic Reasoning (NMR), in order to make decisions even with partial and contradictory information. Secondly, to consider the property of resilience which is particularly important for adapting changes and last the implementation of the model in an embedded computer for an UAV.

Actually, there are UAV that use control methods for doing autonomous purposes. However, they are based on petri nets or differential equations (Guilliard, Rogahn, Piavis, et al. 2018). In both cases, they can not solve the problem of incomplete and contradictory information (Tabor, Guilliard, and Kolobov 2018). This is a very important topic in AI that has been studied for more than 40 years. It should be pointed out that in this thesis we are not considering methods such



(a) Phases of flight



(c) Ideal conditions (d) Real conditions

(d) Principle of navigation

Figure 1.10: Decomposition in phases of a flight mission and Flight plan.

as learning (clustering, neural networks, genetic algorithms...) or controls laws (PID, adaptive control, robust control...) We address this problem from the point of view of AI using NMR, more especially Reiter's Default Logic (DL).

Currently, most of the autonomous systems are based on bayesian networks, petri nets or differential equations. Bayesian networks have two major problems. The first focus on the computational difficulty of searching for unknown information. Since the probability of each solution must be calculated for all branches of the network. This being a problem NP-hard knowing the number of variables and combinations. The second problem focuses on inference. It must be borne in mind that an excessive optimistic or pessimistic mathematical hope of the quality of the inference will alter the network and invalidate previous results (Kolobov 2016; Kapoor, Dey, Kolobov, et al. 2017). Selecting the correct distribution that describes the data has a remarkable quality of the response of the network (Niedermayer 2008). Meanwhile petri nets have limitations in terms of complexity. The models are usually very complex for analysis. In practice it is necessary to make modifications and restrictions for its operation in a particular application

(Murata 1989).

Another manner for modelling systems is to use integro-differential equations or state-space form (Fossen 2011). But, there are no applicable general methods to solve partial differential equations. However, there are methods that can be used for very particular cases. The Split-step method is a pseudo-spectral method for the solution of non-linear partial differential equations, e.g. nonlinear Schrodinger equation. An alternative is the use of numerical methods with finite differences or finite elements. In both cases high performance computers are required (Bellman, Kashef, and Casti 1972). E.g. in a flight of a 787 aircraft could generate a Terabyte of data. Only the engines send information of 5000 parameters. Companies like Boeing, are starting to use AI to interpret this data, and thus be able to have more efficient equipment. The most used techniques are the incorporation of missions previously learned from more than 1000 human flights controlled experiences. Machine learning uses history data and statistical methods to perform future solutions, e.g. regression model.This method is impractical when there are new exceptions. In the most cases, it should be add all the exceptions in the computer on board to be able to cover as most scenarios as possibles. This means that all the exceptions must be known, this can be a very big number.

We are going to approach the problem from the point of view of artificial intelligence. We use default logic, created by Raymond Reiter (Reiter 1980). This is a non-monotonic logic, which has a main characteristic which it allows us to solve situations with partial, contradictory information and exceptions. Exceptions that are the main theme in piloting. Chapter 2 contains everything related to this topic, as well as the model used for decision making. The second objective is about applying the property of resilience to our non-monotonic model obtained, we applied this property thanks to Minksy's model, the theoretical aspect can be found in chapter 3. Chapter 4 addresses the practical side, it explains implementation on the microcomputer, physical, electrical and electronic characteristics of our UAV.

Conclusion

In this chapter we described the parts of an airplane, how it is controlled and we saw the laws that govern it. As well as the mathematical models that can be used for modelling. An analysis of the trajectory was described, which can be summarized in five main phases (Rest, Start, Takeoff, Climb, Bank turn, Steady flight, Descend, Final approach and Land). Eventually, we described the challenges of piloting, like environment without obstacles or without disturbances of the climate (wind, rain, snow...), so a pilot should be able to compensate and control these uncertain effects. Eventually, the motivation of this thesis.

2 Non-Monotonic Reasoning

The objective of this section is to present a formalization of our problem allowing to reasoning under uncertainty, incomplete and contradiction informations. We will see that classical logic has limitations, and we have to move to NMR. For this we present a particular logic which is a means by which we can carry out our problem. After having solved the problem of contradictions and incomplete information, this can have several solutions. For this, an opportunistic notion is taken into account to choose the best option according to the criteria made. Eventually, we will see an implementation in prolog which allows us to calculate the possible solutions in a given situation.

Summary

2.1. Classical Logic

Logic is a particular way of thinking. This allows us to reach conclusions considering certain information to find an answer. But also, logic study the formal principles and systems of inference. It is used in many fields of research such as mathematics, science or philosophy. Nature and the human sciences are based on logical, analytical, systematic and objective thinking to investigate and discover. Our cognitive capacity for logical thinking is based on rationality.

The formal systems of logic, e.g. propositional, predicate, modal, conditional, mathematical..., are symbolic constructions in a particular language designed

to analyze and model modes of inference. For example, deductive, inductive and abductive.

In order for a system to reason, learn or explain its knowledge, it must first formalize its ideas. It is normal that when we learn something first we must know what it is that represents such a thing. So we need to use an appropriate language so that knowledge must be formalized. Next, a specific language is presented for the aforementioned purpose. The language of First-Order Logic, abbreviated FOL. There are other types of logic, however, for its convenient expressiveness, FOL will be used.

A language is a set of words or symbols that represent something. Which has three characteristics. The first one is the syntax. In this we have two types of symbols: logical and non-logical. The logical symbols are those that have a fixed meaning like: the connectors $(\neg, \wedge, \vee, \exists, \forall)$, the last two are called quantifiers; and variables that are a source of infinite symbols, are usually denoted as: x, y, z. Non-logical symbols are those that have a meaning depending on the application. Here we can find the symbolic functions detonated generally as: a, b, c, f, and the symbolic predicates which are denoted as: P, Q, R. The second feature is semantics. It focuses on the meaning of expressions. And the third on pragmatics that deals with how we use the meaning of the expressions of a representation in a language as part of a knowledge base about the world to later make inferences. Expressing knowledge in FOL is very natural. Knowledge is the representation of the world which satisfies certain properties (Moore 1981; Moore 1984). In the next section we will see the limitations of classical logic.

Limits of Classical Logic

In mathematical logic, the set of predicates in a formal language is the component that constitutes the syntax of a theory. A model of a theory is a structure that satisfies the predicates of this theory. FOL is an extension of classical logic. It is a formal system that uses quantifiers such as $\exists, \forall$ over variables. FOL consists of rules describing what valid sentences are and how these rules can be interpreted in models. The principle of second-order logic is the same, only that it has more expressive power. This allows to arbitrarily quantify certain sets in the model. FOL has a set of derivation rules which is complete. This is what Godel's theorems of completeness indicates. Unfortunately for second order logic, this rule set has no derivation. This means that there is no way to capture what it means for one thing to follow another.

A theory is a set of predicates, expressed in a certain language. To give some examples, Peano's arithmetic is a theory, Zermelo-Fraenkel set (ZFC) is also a theory. The theories can be consistent. This means that it does not contain a statement of P and its denial $\neg P$. Also the theories can be complete, meaning that they contain even P or $\neg P$ for each statement of P. FOL is a formal system

(not a theory). A theory is a set of axioms from which consequences can be derived. The set of derived rules is called complete if each semantic consequence is also a syntactic consequence.

Having thus described the principles of FOL, we can start by use it to formalize the pattern circuit. We could say: "An airplane is landing". In logic we have: $land(airplane)$. Another instance could be: "A pilot increase the engine power". In FOL we have: $engine(pilot, increase)$ and so on. These kind of predicates would define the basis ontology of this world. In this way and making use of the flight manual we can represent more predicates, because it contains all necessary information for piloting, including technical descriptions, physical limitations of the airplane, rules and emergency procedures. But all these information are generals that depending on the situation could have contradictory rules. For instance, there is a rule that says the minimum over flight height will never be less than 500 feet, in fact this altitude depends of the agglomeration. This rule could be expressed in FOL, considering that $x =$ airplane, as follows:

$$\forall x, [altitude(x) \rightarrow (x > 500)] \tag{2.1}$$

But when an airplane lands its altitude is less than 500 feet. This could be expressed as follows:

$$\forall x, [land(x) \rightarrow (x < 500)] \tag{2.2}$$

Another more general rule covers the notion about aircraft. That is, it is very well known that aircraft have tires in the landing gear, which just allows them to land and take off. We can formalize this with the following rule:

$$\forall y, [aircraft(y) \rightarrow tires(y)] \tag{2.3}$$

But we also know that floatplanes are aircrafts that do not have wheels. So, we can have the following formalization:

$$\forall y, [floatplanes(y) \rightarrow aircraft(y)] \tag{2.4}$$

$$\forall y, [aircraft(y) \rightarrow \neg tires(y)] \tag{2.5}$$

We can see that formalization (2.1) and (2.2) are contradictory, and formalization (2.3) and (2.5) are too. This is because of classical logic, such as FOL. Classical logic is monotonous. This property is very important in the world of mathematics, because it allows to describe lemmas previously demonstrated. But this property cannot be applied to uncertain, incomplete information and exceptions, as we saw with the predicates previously. This is, by adding new information or set of formulas to a model, the set of consequences of this model is not reduced. This is a limitation of classical logic. We have formally the property of monotony is: $A \vdash w$ then $A \cup B \vdash w$. In other words, adding new information to a model,

the consequences are not reduced.

Remembering what we saw in the Section 1.2. Piloting is an human activity susceptible to having partial information, contradictory situations and exceptions. Because of environment change, maybe emergencies, respect aeronautical and security regulations. Taking all this into account, a pilot must follow his desired flight path. He enters into reasoning under uncertainty. We can see that it is a non-monotonic problem. Especially when we work on Knowledge Representation (KR) and human reasoning. This kind of problem is well known in AI. It has been studied from along time (Minsky 1974; Reiter 1981; McCarthy 1990). In order to tackle it we have to move from this framework of classical logic. Because a pilot uses non-monotonic reasoning when has new information, he verifies them and could break them.

The objective is to represent what is known in FOL and to be able to decide what to do by deducting a conclusion, which will allow to do certain actions. The act of doing actions could create new situations. This brings us to the representation of knowledge about exceptions, contradictory situations, and also of the general knowledge of common sense.

The Need of Non-monotonic Reasoning

The problem leads directly to the general representation of common sense knowledge. We can say that an airplane can fly through the sky unless something prevents it from doing so. The need of NMR is indispensable in this thesis. As we saw in Section 2.1, classical logic has limits, for example the principle of explosion. Moving to another non-monotonic framework we can carry out the explosion principle and nevertheless reach a conclusion. When a pilot goes from the "point A" to the "point B", he has to make decisions at every moment and those decisions come from a NMR where any information can be contradictory and incomplete because of environment, legislation, unexpected situations.... However, a pilot must take actions that allows to reach goals, for example, to determine altitude, to stay for a certain time at a certain altitude, and so on. In recent decades the development and study of UAV has increased in order to extend the flight time. Since a pilot has a certain amount of hours, this is caused by fatigue and can have an impact on the decisions of a pilot. An UAV has the same mechanical and aerodynamic problems as a private or commercial aircraft. All this type of airplanes, they are governed by the same physical principle: sustentation. It is important to note that this study is being carried out in a non-military manner. However, the problem does not change since the UAV must have control of the navigation, trajectory, legislations to respect...

2.2. Non-monotonic Logic

As we previously said, we have to move to NMR framework. KR and human reasoning have a better understanding using this class of reasoning. NMR is a kind of reasoning where we make assumptions about things to jump to conclusions. This type of reasoning is presented in many human situations. That is to say, each new information we learn can modify or invalidate previous deductions. This reasoning began to be studied since the 1970s, J. McCarthy, D. McDermott, Reiter and others were interested in studying non-monotonous inference, resulting in default reasoning, epistemic reasoning and more (El-Azhary, Edrees, and Rafea 2002). One of the most studied and has had good results in practical terms, has been the default logic, proposed by Reiter (Reiter 1980; Reiter 1981). The advantages of non-monotonic reasoning are to increase applications in real domain and to have greater freedom in order to learn things. To formalize this type of reasoning, we will present in more detail why it is important to use this logic in this thesis.

Non-Monotonic Logic (NML) allow us to tackle the principle of explosion. This principle is present when you have in a knowledge an information and the denial of the same information, but nevertheless you can conclude a thing. There are different types of logic that treat this principle, one of them is the paraconsistent logic that was developed in the 2000s. It has different uses in different areas, to describe behaviors in psychology, economics and also in other sciences (Béziau 1999). We know that FOL is expressive, which allows us to describe almost anything in a natural way. FOL uses objects, relations or predicates, functions, connectivities and quantifiers. Let's imagine that we have a world of a certain theme and it is represented in FOL. And we would like to know something in particular about that world. But we can find ourselves with the problem that what we are looking for is an exception. This means that within the world certain characteristics are met but not all, which will prevent us from concluding anything. Assuming we find the particular thing, this can have two explanations. The first is that the world contains explicit facts that constant what we are looking for. And the other explanation is that there is a universal way to conclude our search. Airplanes have engines. Say that airplanes have three wheels. This makes us think of some kind of aircraft. Therefore it is not possible to say that all airplanes have three wheels. One solution could be to list all the exceptions in which the airplanes do not have three wheels: *All the airplanes that are not* P_1 *or* P_2 *or* ... *or* P_j *have three wheels.*, Where P_j are all the particular cases. Then the problem is to know those cases and be able to cover at least the variants such as commercial aircraft, transatlantic, helicopters, floatplane... those that land on the water or snow, those who land on an aircraft carrier.... Due to the large number of cases, it is for this reason the importance of distinguishing between a universal form, properties for all instances, and a generic one, general properties. Much of our common sense knowledge of the world appears to be

contained in a general way. This is why it is important to consider a formalism in FOL that allows us to capture general, but not totally universal, of knowledge.

In real life, there are exceptions where a pilot must land on highways, lakes or aircraft carrier if we talk about the army. In all the above cases, a pilot must make a decision to land. It should be noted that given this situation there will be contradictions, since generally an airplane lands on a runway. However, exceptions may occur and the aforementioned rule can not always be met. This is the case when an airplane must land on a highway, parking lot and even on the sea or lake. What can we say about the following statement: Normally, aircraft have tires that allow them to take-off and land. But an helicopter is an aircraft that has no tires. Climatic conditions, air traffic and even agglomeration are some of the elements that make piloting an activity where NMR is used. A pilot must consider all these aspects that are changing. The weather can hinder the visibility, the airplane can suffer turbulences due to the temperature and atmospheric pressure of the area. Equally important the air-traffic, a pilot is always in communication with the control tower, since each airplane must respect the area regulations, Figure 2.1. As described in the Section 2.1 that piloting, flight

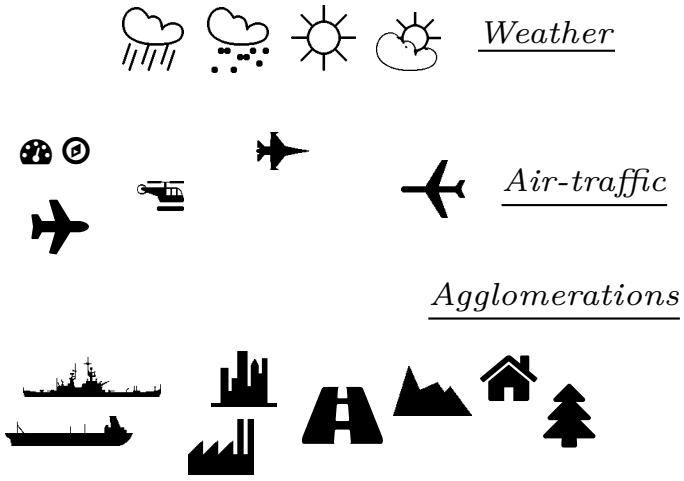


Figure 2.1: Flight environment.

environment... are governed by rules. For example, from the Federal Aviation Agency (FAA) the rule 91.7 dictates that: "*No person may operate an aircraft unless it is in an airworthy condition*" and "*Pilot-In-Command (PIC) is responsible for determining whether that aircraft is in condition for safe flight*". Rule 91.137: "*No one can operate an aircraft in the designed area unless aircraft is carrying event personnel*". Rule 91.319: "*Operate under Visual Flight Rules (VFR), day only, unless otherwise authorized*". Well, we just saw some rules of piloting which are very general. Below we will discuss the subject of default reasoning, which allows us to reason in the presence of general information.This is a type of reasoning

in which conclusions are reached based on what is commonly true. Assuming that airplanes usually land on land. Under what circumstances is it possible to conclude that an airplane that touches the water (lake, sea, channel ...) is an airplane? It will depend on the circumstances because particular cases are infinities. When an airplane is not prepared to land on water but for some reason it should land on water, he will do it.

2.3. Default Logic

It is one of the best known formalizations for default reasoning, created by Raymond Reiter. Which allows to infer arguments based on partial and contradictory information as premises. In Reiter (1980) a default theory is a pair $\Delta = (D, W)$, where D is a set of defaults and W is a set of formulas in FOL. A default d is: $\frac{A(X):B(X)}{C(X)}$, where $A(X), B(X), C(X)$ are well-formed formulas. Where $X = (x_1, x_2, x_3, \ldots, x_n)$ is a vector of free variables (non-quantified). $A(X)$ are the ***prerequisites***, $B(X)$ are the ***justifications*** and $C(X)$ are the ***consequences***. Intuitively a default means,"if $A(X)$ is true, and there is no evidence that $B(X)$ might be false, then $C(X)$ can be true". When defaults are used it means extensions are calculated and we can have zero, one or more extensions. An extension can be seen as a set of beliefs of acceptable alternatives.

2.3.1. Extension

An extension is a possible set of knowledge about the world. Formally, an extension of a default theory Δ is a set E of logical formulas Reiter 1980 with the smallest set that must verify the following property: If d is a default of D, whose the prerequisite is in E, without the negation of its justification is not in E, then the consequent of d is in E.

Definition 1. *Let $\Delta = (D, W)$, an extension E of Δ is define:*

- $E = \bigcup_{i=0}^{\infty} E_i$ *with:*
- $E_0 = W$ *and,*
- *for $i > 0$, $E_{i+1} = Th(E_i) \cup \{C(X) \mid \frac{A(X):B(X)}{C(X)} \in D,\ A(X) \in E_i, \neg B(X) \notin E\}$*

Where $Th(E_i)$ is the set of formulas derived from E_i.

There are two types of implications, skeptical and credulous. A skeptical reasoning is whether a formula deduced in a default theory is deduced by all its extensions. And a credulous reasoning is if a formula deduced in a default theory is deduced by at least one of its extensions (Berzati, Anrig, and Kohlas 2002). We can have a normal default theory when a default follows the form: $\frac{A(X):C(X)}{C(X)}$.

The main characteristic of having a normal default theory is that at least one extension is always guaranteed. The original version of the definition of an extension is difficult to compute in practice. Because $\neg B \notin E$ supposes that E is known, but E is not yet calculated. In the case of normal defaults, E is an extension of Δ iff: we replace $\neg B(X) \notin E$ by $\neg C(X) \notin E_i$. Another variant that we can have is a semi-normal default theory, which has the following form: $\frac{A(X):B(X)\wedge C(X)}{B(X)\wedge C(X)}$, Etherington (1987) demonstrates the conditions to prove the existence of extensions. In this thesis, we use the normal form to represent piloting rules. Because there is a case where the extension calculation can be null, so no extensions. In this manner, we can have the guarantee of being able to perform actions even when we have an extremely unusual situation, because actions are contained in extensions.

Example 1. *Using default logic, from formalization (2.1) and (2.2), we have:*

$$d_1 = \frac{((altitude(x) > 500) \wedge roll(x, stable)) : steady\_flight(x)}{steady\_flight(x)}$$

$$d_2 = \frac{((altitude(x) \leq 500) \wedge roll(x, stable)) : land(x)}{land(x)}$$

$$d_3 = \frac{(land(x) \wedge obstacle) : climb(x)}{climb(x)}$$

In natural sense, d_1 describes if x has an altitude more than 500 feet with a stable roll, and it is possible that x is in a steady flight, then x is in a steady flight. Default d_2 describes that if x has an altitude less than 500 feet with a stable roll, and it is possible that x lands, then x lands. And default d_3 describes if x lands and there is an obstacle, and it is possible to climb, then x climbs. Now, we are going to use these three defaults assuming that we have the following information:

$$W = \{(alt(x) \leq 500), roll(x, stable), obstacle\}$$

From $\Delta = (D, W)$, we calculate the set of extensions. We find $E_1 = W \bigcup land(x)$, where x lands, by using the default d_2. On the other hand, we find $E_2 = W \bigcup climb(x)$, where x climbs, by using the default d_3. We have two coherent solutions. Solving the problem of contradictory information.

There are mandatory rules that cover flight physics, security and more. For instance, in case of engine failure, x lands. Or if there is an obstacle in the runway, x must not land. But if x has an emergency, a pilot must land to not die, so the risk is enormous. The mandatory rules will have a high weight, because of priority and security. When different solutions are computed we should take into account criteria such as emergency, security, regulation, energy... In the next section we are going to describe the different manner to choose an option of a multiple alternatives. The chosen one will be the one that complies with the imposed restrictions function.

2.4. Decision Making under Uncertainty

Decision making problem is one of the most important field in multiple criteria decision (Triantaphyllou 2000). It exists many methods such as: Pareto analysis, decision matrix analysis, prospect theory... In this section we present a non-probabilistic model to decision-making under uncertainty. In real life, we interact in a non-deterministic environment that cannot be really described by probabilistic or mathematical models. This is because every situations could change over time. Decision-making is a problem that everyday we do. Let's take an example, when we decide to go buy orange juice from your home to the market. Let's think you have enough money to buy orange juice, so you are ready to go at the market place by walking. Before you close the door of your apartment, you don't know what is the next situation, maybe the elevator is not working and you need to use the stairs or maybe the street that you usually use is closed for repairs. It should be noted that, as can be seen in these two examples, non-monotonic and non-probabilistic decision-making is presented. Because of non-monotonic world, we have also non-probabilistic decisions. We can not be sure that if you use the elevator 20% of the chances it will not work, or 6% of the chances street will be close for repairs. In this part there are different methods to use, since we have a problem of optimization multi-criteria (Sion 1958; Chiles and McMackin 1996; Cvetkovic and Parmee 2002; Moshman 2004; Park, Johnson, and Kuipers 2012; Shieh, Li, and Yang 2017).

- **A priopri:** Decision $\rightarrow$ Optimization.
- **Progressives:** Decision $\leftrightarrow$ Optimization.
- **A posteriori:** Optimization $\rightarrow$ Decision.

Decision-making is done in a non-probabilistic manner, in others words, when a person must make a decision he does not have to calculate a distribution. The problem is not trivial because we have to perform a function that minimizes the risks and maximizes the benefits like that we can find an optimal option which will converge according to a goal. Since the extensions contain actions that will be applied.

2.4.1. Probabilistic

This kind of decision-making which is based on probabilities for each alternative. In this model we have a previous quantification of the samples for each case. Therefore, a number of experiences must be performed, statistically 1000 times, to have a considerable percentage of certainty. Considering the following Table 2.1, where the alternatives can be appreciated, in our case they are fixed points found, and their respective weights and probabilities.

Alternatives	Criteria		
	0.4	0.2	0.4
d_2	4	4	5
d_3	3	3	2
d_4	3	2	3

Table 2.1: Weighted criteria.

A commonly decision-making used is called weighted product. In which is defined by the following notation:

$$P(A_K/A_L) = \prod_{j=1}^{n} (a_{Kj}/a_{Lj})^{w_j} \tag{2.6}$$
$$For\ K, L = 1, 2, 3, \ldots, m.$$

Applying the Equation (2.6) to the Table 2.1, we will obtain the following results, to compare d_2 with d_3:

$$P(d_2/d_3) = (4/3)^{0.4} \times (4/3)^{0.2} \times (5/2)^{0.4} = 1.71443 > 1$$

To compare d_2 with d_4:

$$P(d_2/d_4) = 1.58089 > 1$$

To compare d_3 with d_4:

$$P(d_3/d_4) = 0.92213 < 1$$

In order to make a comparative decision, the result of the operation is compared with the unit, this comparison is empirical. If the result is greater than unity, it can be concluded that d_2 is better than d_3, ($d_2 > d_3$). Otherwise, if the result is less than the result, we can conclude that it is not the best option, as it can be shown in the next result. So, using the model of the Equation (2.6), we have that $d_2 > d_3$, $d_2 > d_4$ and $d_4 > d_3$. Concluding that d_2 is the better option, in second place d_4 and the last place d_3 ($d_2 > d_4 > d_3$).

2.4.2. Non-Probabilistic

The advantage of being able to work with non-probabilistic models is that it reduces the quantity of complex operations, for example power operation is not necessary, as we saw before. Another advantage is that it is not necessary to make experiences of the alternatives. The principle is very simple, we ponder all the solutions in order to put priority, in our case are defaults. In other words, a default which has a higher value than the others, it is the most important. We

will start by defining a vector that goes from the lowest to the highest priority, Table 2.2. This can be seen as a scale of importance, priority, risk,... For non-probabilistic models, the aforementioned table will be taken as a reference for the others three types of criteria that do not use probabilities. The first model

Score				
Very low	Low	Medium	High	Very high
0	1	2	3	4

Table 2.2: Criteria scale.

presented is called **MaxiMax**. The idea is to find the best of the best for each case. In other words, it would be choose the best possible and then take the actions with the highest values, this model is also called optimist. If we calculate three

Alternatives	**Criteria**			**Best**
	A	***B***	***C***	
d_2	1	0	1	1
d_3	4	2	4	**4**
d_4	3	2	3	3

Table 2.3: Maximax criteria.

alternatives, as defaults with different criteria and ponderation. For MaxiMax, the ***best choice*** is d_3, because for each alternative we maximize the values and then we take the maximum value of them, Table 2.3.

For the second model, **Maximin**, we will choose the best of the worst alternative for each case, this model is also called pessimist. With the same alternatives

Alternatives	**Criteria**			**Worst**
	A	***B***	***C***	
d_2	1	0	1	**1**
d_3	4	2	4	4
d_4	3	2	3	3

Table 2.4: Maximin criteria.

and criteria as before, but now for this model, the ***best choice*** is d_2, because we maximize the values of alternatives and then we take the minimum value of them, Table 2.4. And finally, **Minimax**, for this model is necessary to make an intermediate calculation before choosing an alternative. This intermediate stage is called opportunistic loss (or regret) matrix. Formally, the set of regrets is defined as:

Definition 2. *For each set of calculated extensions, there will be an opportunistic matrix.*

$$\forall E, \exists\ m_r = min\ \{max\ (c_i) - c_j\}$$

Where m_r is the minimization of the difference between the maximum value of the criteria c_i and alternatives c_j, for all extensions. To calculate the matrix of alternatives, the following formula must be followed: *regret* (R) = *payoff received* (P) - *best payoff* (B).

$$R = P - B$$

For each criteria/situation, *regret* = *highest value* - *smallest value*. Then, we choose the maximum value, and finally we choose the minimum of the maximum value. This will allow us to know what can happen before the event happens and choose the best alternative.

Alternatives	**Criteria**		
d_2	1	0	1
d_3	4	2	4
d_4	3	2	3
Regret table			
d_2	4-1 = 3	2-0=2	4-1=3
d_3	4-4 = 0	2-2=0	4-4=0
d_4	4-3 = 1	2-2=0	4-3=1

Table 2.5: Minimax criteria.

Alternatives	**Criteria**			**MiniMax**
d_2	3	2	3	3
d_3	0	0	0	**0**
d_4	1	0	1	1

Table 2.6: Regret table.

For the same alternatives and criteria, for this model the ***best solution*** is d_3. First it should be calculated the regret/loss matrix, Table 2.6, second, we maximize the values and then we take the minimum of them. In this thesis, we use this model due to its opportunistic characteristic. Because it creates an operational space where it shows us the losses, before event happens and then choose the best option. In this manner, we optimize in the opportunist way and then we choose a decision, referring to an “a posteriori" decision making. The biggest advantage of this model is that it has more flexibility in terms of predicting events,

precisely because of the calculation of the matrix. We formally define our model of choice of extensions.

Definition 3. *For each default (d) there is a weighting (p). When the extensions are calculated, the one that satisfies the opportunistic notion will be chosen.*

Proof. Knowing that each rule has a weight, $d_x = \{0, 1, 2, \cdots 100\}$. The set of solutions is $E = \{d_{x1}, d_{x2}, d_{x3}, \cdots\}$, each rule containing certain criteria, $E = \{[x_{d1}, y_{d1}, z_{d1}], [x_{d2}, y_{d2}, z_{d2}], [x_{d3}, y_{d3}, z_{d3}], \cdots\}$. We must normalize for each criterion:

$$\begin{gathered}\frac{C1_n}{x_{d1}} + \frac{C2_n}{x_{d2}} + \frac{C3_n}{x_{d3}} + \cdots \in |C1_n| \\ \frac{C1_n}{y_{d1}} + \frac{C2_n}{y_{d2}} + \frac{C3_n}{y_{d3}} + \cdots \in |C2_n| \\ \frac{C1_n}{z_{d1}} + \frac{C2_n}{z_{d2}} + \frac{C3_n}{z_{d3}} + \cdots \in |C3_n| \\ \vdots\end{gathered}$$

Creating standardized weights for each solution: $E_n = \{|C1_n|, |C2_n|, |C3_n|, \cdots\}$, $E_{n-1} = \{|C1_{n-1}|, |C2_{n-1}|, |C3_{n-1}|, \cdots\}$, $E_{n-2} = \{|C1_{n-2}|, |C2_{n-2}|, |C3_{n-2}|, \cdots\}\ldots$ We obtain the following standard table of the solutions with their criteria: Applying

Ext-Crit	$C1$	$C2$	$C3$	$\cdots$
E_n	X_n	Y_n	Z_n	$\cdots$
E_{n-1}	X_{n-1}	Y_{n-1}	Z_{n-1}	$\cdots$
E_{n-2}	X_{n-2}	Y_{n-2}	Z_{n-2}	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 2.7: Normal extension-criteria table.

the opportunistic principle which in theory of the decision leads us to consider the minimax function, we find the solution, E_n. □

2.5. Study Case

Isabel's thesis is the first research where extensions are calculated in prolog language, her thesis was about *"Modelling of Submarine Navigation"* (Isabelle Toulgoat, Botto, De Lassus, et al. 2009; I. Toulgoat 2011; I. Toulgoat, P. Siegel, and A. Doncescu 2011; Isabelle Toulgoat, Pierre Siegel, and Lacroix 2011). Later, this application inspired to use it in the study of biological systems (Le, A. Doncescu, and P. Siegel 2013; A. Doncescu and P. Siegel 2015), and now for piloting.

In this section, we are going to present the representation of defaults D and facts W for our case. Default set D embeds actions that will take into account: exceptions, partial and contradictory knowledge. Accurate informations and mutual exclusions are in W. Mutual exclusions are actions that can not happen at the same time. All the information of the pilot and airplane will be described inside of two sets. For example, this can be like "a pilot and an airplane are on runway" and also "a pilot does a checklist". The goal is to derive conclusions, they are expressed such as defaults. To takeoff three defaults are described:

$$d_0 = \frac{(authorization(pilot, takeoff) \wedge \neg obstacle) : motor(pilot, on)}{motor(pilot, on)}$$

$$d_1 = \frac{motor(pilot, on) : motor(pilot, max)}{motor(pilot, max)}$$

$$d_2 = \frac{airplane(airspeed) > 1000 : yoke(pilot, pull)}{yoke(pilot, pull)}$$

This happen when we assume that a pilot is in the airplane and he has the authorization to takeoff, which was sent from the control tower. Otherwise, we can conclude that he will wait for the authorization signal. For the following flight phases, general rules are also presented, for the next states such as climb:

$$d_3 = \frac{airplane(airspeed) > 1000 : yoke(pilot, pull)}{yoke(pilot, pull)}$$

General rule for bank-turn:

$$d_4 = \frac{airplane(altitude) > 1000 : yoke(pilot, left)}{yoke(pilot, left)}$$

$$d_5 = \frac{airplane(altitude) > 1000 : yoke(pilot, right)}{yoke(pilot, right)}$$

Motor	Pitch
$motor(pilot, on)$	$yoke(pilot, push)$
$motor(pilot, half)$	$yoke(pilot, neutral\_p)$
$motor(pilot, off)$	$yoke(pilot, pull)$
Roll	**Yaw**
$yoke(pilot, left)$	$rudder(pilot, left)$
$yoke(pilot, neutral\_r)$	$rudder(pilot, neutral)$
$yoke(pilot, right)$	$rudder(pilot, right)$

Table 2.8: Actions in FOL.

General rule for steady-flight:

$$d_6 = \frac{airplane(altitude) > 1000 : yoke(pilot, neutral)}{yoke(pilot, neutral)}$$
$$d_7 = \frac{airplane(altitude) > 1000 : motor(pilot, half)}{motor(pilot, half)}$$

General rule for descend:

$$d_8 = \frac{airplane(descend) : yoke(pilot, push)}{yoke(pilot, push)}$$

General rule for final-approach:

$$d_9 = \frac{airplane(variometer) < 0 : yoke(pilot, neutral)}{yoke(pilot, neutral)}$$

General rule for landing:

$$d_{10} = \frac{airplane(altitude) < 10 \wedge \neg obstacle : motor(pilot, off)}{motor(pilot, off)}$$

A pilot can apply different actions: engine power, elevator, ailerons and rudder. These actions are defined by the following predicates, Table 2.8. It should be noted that mutual exclusion must be considered. Since in our model it can not be true $yoke(pilot, push)$ and $yoke(pilot, pull)$, because it is physically impossible. They are considered as positive literals. For example, given a situation, only four predicates can be true and these must be different.

2.5.1. Implementation in Prolog

The implementation to calculate the extensions is done in Prolog. Prolog is a powerful and flexible programming language. This language means “logical pro-

gramming", but in french "PROgrammation LOGgique". Among its main tools is the recognition of patterns, tree structure for data and automatic regression. It was specially made to solve problems involving objects and the relationships between them. The ease with which you can work in this language is very powerful, both in Artificial Intelligence and in non-numerical programming. We can have the following rules as an example: if X is closer to the observer than Y and Y is closer than Z, then X is closer than Z. Thus prolog can reason about spatial relationships and their consistencies based on general rules. This idea emerged in the 70s when it began to use logic as a programming language. Some people were developing the theoretical aspect as Robert Kowalski in Edinburgh, Maarten Vaan Emden also in Edinburgh and the experimental side Alain Colmerauer in Marseille. Eventually it evolved but became popular when David Warren made a very efficient implementation in Edinburgh in the mid-70s.

We start to represent the information of the cockpit, $Glider(x)$ where x are the information of the sensors. For instance, airspeed is represented with three informations, using prolog syntax we have: "Airspeed([low,stable,high])", low indicates that the measure is less than a value, $stable$ is between two values and $high$ indicates more than another value, analogy for: "Altimeter([low,stable,high])". "Pitch([up,center,down])", up describes a positive angle of attack, $center$ is no inclination and $down$ describes a negative angle of attack, the same way but vertical speed: "Variometer([up,stable,down])". "Roll([left,center,right])", $left$ indicates turing to the left, $center$ is no turning and $right$ indicates turning to the right, analogy for: "Turncoord([left,center,right])" but tangential force. Finally, "Compass([n,s,e,w])", *n,s,e,w* indicate the direction of the airplane to the north, south, east and west, respectively.

2.5.2. Facts and Rules

In Prolog there are two important components: the facts and the rules. The facts are a collection of truths in the domain, and usually they are ground atoms; The rules are used to extend the vocabulary, expressing new relationships based on basic facts, usually containing quantifiers.

$$
\begin{aligned}
&runway(airplane1, cabin(pilot1)).\\
&runway(airplane2, cabin(pilot2)).\\
&authorization(cabin(pilot1), airplane1).\\
&\ldots\\
&takeoff(x, y) \leftarrow runway(x, y)\\
&takeoff(x, y) \leftarrow authorization(y, x)\\
&flying(y, x) \leftarrow takeoff(x, y)\\
&\ldots
\end{aligned}
$$

As we can see, the rules: $takeoff(x, y) \leftarrow authorization(y, x)$ can be read as "x take-off for y if y has the authorization to take-off x". This rule will be true if the right side is true. From the facts we know that there are two airplanes in the runway and "$pilot\ 1$" of the "$airplane\ 1$" has authorization to take-off. The first rules is to know if the airplane is located in the runway, and if it is true and authorization too, airplane take-off, therefore it will fly. Unless another facts invalidates this rule. To find the solutions to our system it is necessary to solve a set of logical equations. Computational difficulties are the biggest challenge in this type of case. However, with the use of the horn clauses, complexity becomes more manageable and the expressiveness suffices for many purposes. Clauses are the simplest type of formula. Formally, a clause is a disjunction of literals $l_1 \vee l_2 \vee l_3 \vee \ldots \vee l_n$. A Horn clause is a clause with a maximum of one positive literal. It's a formula defined as $(f_1 \wedge f_2 \wedge f_3 \wedge \ldots \wedge f_i) \rightarrow g$, where f_i and g are positive literal. Similarly, the formula defined as $\neg(f_1 \wedge f_2 \wedge f_3 \wedge \ldots \wedge f_i)$ is equivalent to the negative Horn clause(literals can not be true simultaneously). An example of mutual exclusion: $\forall t, \neg(glider(pilot, yoke\_left) \wedge glider(pilot, yoke\_right)$. When Prolog is working in a proof, it will consider false what he can not prove true, and vice versa. This is known as the theory of closed worlds. In simple terms, the negation of a predicate does not imply that all time is false, simply that it is not demonstrable with the information given. This principle is called Closed World Assumption, it means that everything that is not known is considered false (Reiter 1981).

For our purposes, we define clauses as rules of piloting. These rules follow a specific syntax which has the form: $cl(\texttt{text}, \delta, A, C, \omega)$. Where `text` could be a chain of characters as comment, δ could be a real fact "`hard`" or default "`def`", A and C are the prerequisite and consequent, and finally ω is the priority weight and could be a vector of criteria, Table 2.7. The theoretical computational complexity is $\sum_2^p$. In our implementation normal defaults and Horn clauses are used only. Allowing to compute the extensions in a quasi-linear time. However, if we increase the number of defaults, the calculation time does not increase exponentially, others applications (Le, A. Doncescu, and P. Siegel 2013; A. Doncescu and P. Siegel 2015). An algorithm for the calculation of extensions is presented. It is based on normal defects, this guarantees the calculation of at least one extension.

Conclusion

In chapter 2 was described how the problem was studied. We start by modeling basic rules of piloting by FOL. But the representation of the rules and environment can not be solved by classical logic. Piloting rules are uncertain and contradictories and they have to verify them at every time. Non-monotonic logic has a great advantage in these kind of cases. It is an important tool in Artificial

Algorithm 1 Calculation of extensions.

Require: $E = \emptyset$
  procedure EXTENSION($E = \cup_{i=0}^{N} E_i$)
    Initialization
    CalculExtension(E)
    while there is a default $\frac{A(X):C(X)}{C(X)}$ that has not yet been inspected **do**
      Select the default D
      Verify A(X) are true
      Verify C(X) is consistent with W
      Add C(X) to W
    end while
    Backtracking(deleting C(X) added to W)
    CalculExtension(E)
  end procedure

Intelligence because it captures reasoning under uncertainty. In our approach, an algorithm based on default logic is proposed. We did a simulation of take-off situation and we found 5 extensions that are consistent subsystems. When we have two or more solutions we must choose one of them. We ponder all the defaults depending on security, priority, legislation... Decision making is based on an opportunistic notion. Which makes a non-probabilistic analysis of the possible scenarios before they occur (Sion 1958; Chiles and McMackin 1996; Triantaphyllou 2000). The results provide the basis for an autonomous motor-glider. The theoretical computational complexity is $\sum_2^p$. Normal defaults, Horn clauses and normal criteria weights are used only. Allowing to compute the extensions in a quasi-linear time. We presented short and simple Prolog programs. For initial tests(around 100 rules), we calculate all extensions in milliseconds. The average calculation takes 1956600 Logical Inferences Per Second (LIPS) and 0.049 seconds of central processing unit (CPU) time on a MacBook.

3 Resilience

The objective of this section is to present the definition and properties of resilience. We will see the KOSA model, which is a model which contains a set of knowledge described in objectives, situations and actions. To make a better capture of the property of resilience in our model based on Reiter's defaults logic, we will define concepts such as short and long objectives. We will make use of Marvin Minsky's theory, as well as the definition of new concepts. Finally, we will present the discrete and continuous behavior of the total model.

Summary

3.1. Definition

This thesis arises from the concern of being able to capture the notion of resilience using a language that allows to represent any knowledge in a natural way. But first, what is resilience? It is the property of a system that has the ability to absorb and to adapt in the presence of disturbances. Crawford S Holling (1973) introduced the term of the resilience to model the dynamics of natural disasters (Chandra 2010; S. Goerger, Madni, and Eslinger 2014). In other fields of science the concept of resilience is defined as the property of a system to absorb and anticipate perturbations. In ecology, resilience aids to understand natural disasters behavior. In engineering, resilience ensures consistency, robustness and stability (Sitterle, Freeman, S. R. Goerger, et al. 2015), even in uncertain environments (Zhang, Mahadevan, Sankararaman, et al. 2018), or can be defined as safety management which concentrated on no matter what situation

under pressure, it will converge successfully (Crawford Stanley Holling 1996; Woods and Hollnagel 2017). In psychology is the overcome a personal difficulty or stress' life, (Fletcher and Sarkar 2013; McEwen, Gray, and Nasca 2015). In economy is a moment in which there is a balanced situation of diverse variables.

When a system suffers disturbances, internally its states are reconfigured. These modifications can cause exceptions and contradictions, this can be seen as a disorder of the states. This disorder will have a behavior in the face of the magnitude of the shock witnessed. If the system is sufficiently resilient, it can absorb shocks easily. In the worst case, it can collapse because it is not very resilient. Once the system has absorbed the shock, it will resume its previous behavior.

The concept and definition of resilience was introduced by Crawford S Holling (1973) to describe behaviors of ecological systems, studying the stability of them. Resilience is the property of a system of persisting. This being a quantitative characteristic to absorb changes of variables and parameters and continue to persist. On the other hand, stability is the ability of a system to return to its equilibrium point after having suffered a disturbance. It is said that a system is very stable when it returns faster to its equilibrium point, and has fewer fluctuations. Eventually, he describes resilience like adaptatives cycles (Crawford S Holling 2001).

When a system is affected by a constant external changes, it is confronted to the unexpected. In most of the cases, it will find a stable point then constancy of its behavior is less important than its persistence. To better explanation, let's give an example. We can imagine the following drawing where we have a ball and different local minimums. If we apply a force to the ball, the ball will find another local minimum different from the initial one. However, it will depend on the surface. If the ball is on an almost flat surface, it will be very easy to find another local minimum. Otherwise if the surface has a concave surface, it will make very difficult for the ball to find another local minimum. The fact that the ball is difficult to get out of its local minimum means that it is very stable, since any force on it will have no effects. In contrast, if the ball is affected by some external force where it makes leave the original local minimum and find another, this indicates that the ball is unstable. In the Figure 3.1, field vectors represents a behavior, depending on initials conditions we will have different solutions.

The stability gives emphasis to the balance, maintaining a predictable world and keeping the least number of fluctuations before external disturbances. Having a management based on resilience, this can maintain the options. Resilience implies persistence, so it does not require an accurate capacity to predict the future. The qualitative capacity is sufficient for the system to absorb and adapt to future events for any unexpected way.

In control theory, a very well known way to study the stability in dynamics systems is Lyapunov function. This function has a point of equilibrium x_o of a homogeneous equation $\dot{x} = f(x)$ will be stable if all the solutions of the equation

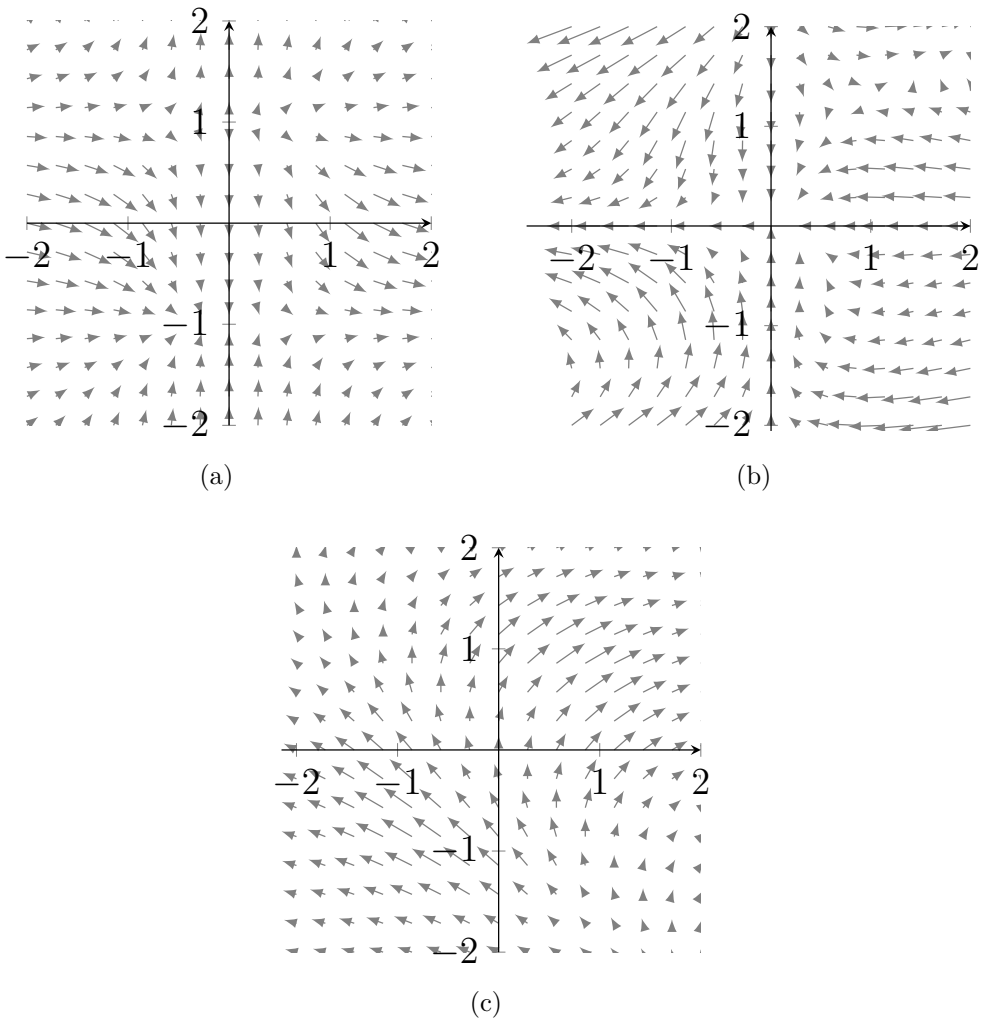
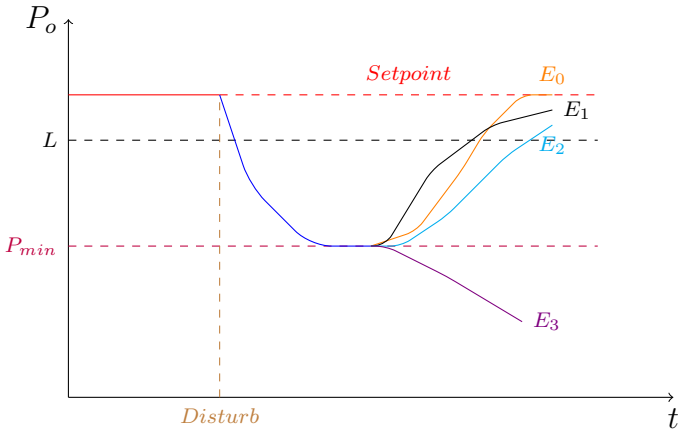
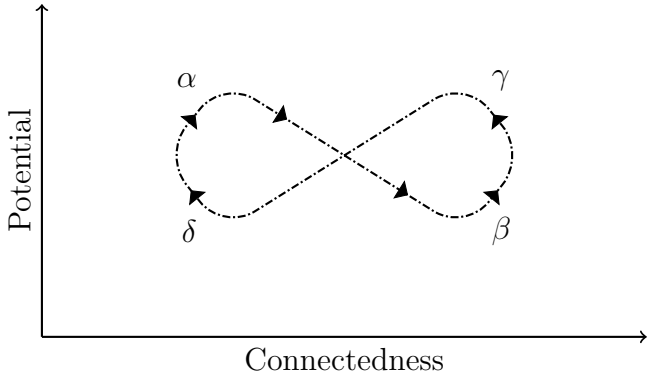


Figure 3.1: Systems with different behaviors.

are found in a environment of x_o and remain in x_o for all subsequent time. In mathematics, it is a condition in which disturbances in a system do not produce drastic effects on it. Lyapunov function involves differential equations which are not of much interest in this thesis. When the system suffers disturbances, it will respond with a behavior that will be defined by many internal factors that determine the reaction time, shock absorption, change of objectives... Figure 3.1. Figure 3.2 shows that at the time of a disturbance the internal states of the system will enter a reconfiguration zone, P_{min}, this can be like the calculation of the extensions from our point of view in DL. Therefore, when the system has a solution, it will begin the recovery process ,$Set - point$, towards the objective it had before the collision. A possible behavior before perturbation could be (a) from Figure 3.1. After perturbations and depending on what extensions chosen $E_0 \ldots E_3$, could result such as (b) or (c).



(a) Response over time



(b) Internal states

Figure 3.2: Resilient system.

3.1.1. Adaptive Cycles

In the theory formulated by Crawford S Holling (2001), resilience is defined such as adaptive cycles too. These are three properties: potential, connectedness and adaptability. Potential determines future solutions, it also determines the limits of what is achievable. Connectedness is the degree of connectivity between internal variables and processes, this is a measure that can describe the level of flexibility or rigidity of the control (sensitivity or not to disturbances). And finally, the ability to adapt, this is the resilience of the system, reflecting its vulnerability to an unpredictable or unexpected event.

Resilience can be formally described under three principles which were already addressed. To explore the solutions of a given situation, we calculate extension thanks to a default theory, Section 2.3, remembering that it is important to take into account the use of normal defects. The second principle is the opportunistic normal function for the choice of an extension, Section 2.5. We must also include the Minsky's model, thanks to this model we can determine a convergence hori-

zon. This horizon can be short or long according to the current situation. In the same way, this model will allow us to know the orbit in the world of knowledge, as well as its behavior, Section 3.2.5. With the set of these principles, we can describe the property of resilience. The major contribution of this thesis focuses on a model that is presented below.

3.2. KOSA Model

The proposition of a model based on a non-monotonic logic and resilient is called KOSA model. As we presented in Section 1.2, the problem is reasoning under uncertainly to follow up different objectives. Here, the problem lies about piloting an airplane where several criteria must be taken into account, the height of the flight will depend on the agglomeration of the area, indicated by the FAA, criteria may include priorities, risk, security, legislation... This is what a pilot has such as information/situation, Figure 1.9. The representation for each flight states in default logic, these are general rules. Firstly, a pilot has to do the checklist to be sure everything is good, according to the regulation.

Definition 4. *Two predicates are defined in the world* $\boldsymbol{K}$*. The first one describes the properties of the airplane and situations.*

$$F(G) \supseteq S, O$$

Where F *are the information of the cockpit, situations... and* G *is a vector of free variables. Some examples could be that the cockpit of the airplane shows an altitude of 1000 feet,* $altitude(glider, 1000)$*. the cockpit of the airplane shows an airspeed of 80 Knots,* $airspeed(glider, 80)$*, the cockpit of the airplane shows a vertical speed of -10 feet/min,* $variometer(glider, -10)$*, but also it can describe the situations, for instance, from the control tower the glider has authorization to take-off,* $authorization(glider, takeoff)$*, glider has an obstacle in front of him,* $obstacle(glider)$*, glider has an emergency,* $emergency(glider)$ *and so on. And the second predicate defines all the actions that a pilot can do.*

$$P(Q) \supseteq A$$

For instance, a pilot turns the yoke to the left, the follows sentences could be described as $do(pilot, yoke, left)$*, a pilot pulls the yoke,* $do(pilot, yoke, pull)$*, a pilot turns on the motor,* $do(pilot, motor, on)$ *and so on.*

The KOSA model is a generalization for the representation of resilient systems. This model is composed of a NML based on DL, non-probabilistic decision making and the Minsky's model. The latter will be explained in detail later in this section. This is a general model that can be used not only for piloting an airplane. But also in all system that human reasoning is involved. From solving

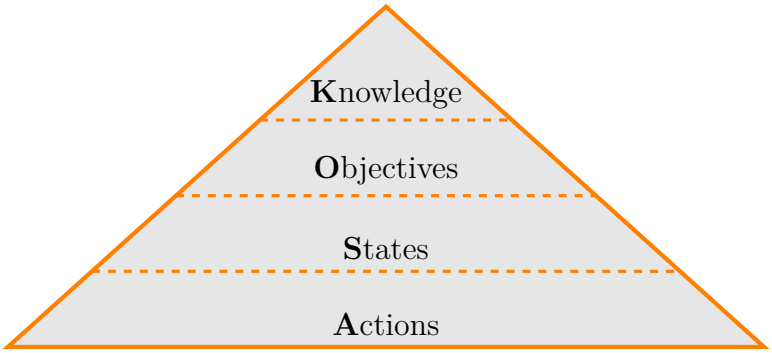


Figure 3.3: Architecture KOSA.

problems like playing chess to driving a car. Because both in chess and driving a car, decision-making under uncertainty is always present. In a game of chess, a player "A" guess the strategy of player "B" to win. Driving car is more complex because the environment change, there are more drivers, we can see that such as more players.

All the systems have an objective as a goal, when new information becomes, maybe a shock, new situations are present. For example, certain biological systems controlled the level of x chemical concentration. Or power electric system controls the voltage required for a good operation of the circuits, e.g. in a motherboard, cellphones, electric cars... But we never know when a shock or disturb will happen. These examples before mentioned are resilience, biological systems can naturally regulate chemical concentration to maintain alive y number of cells. In power electric system, regulation of voltage or current generally done by PWM. The principle of PWM is that frequency changes when impedance of the load changes, like that the level of Root Mean Square (RMS) of voltage or current in the load connected is maintained.

To illustrate the notion of the model, in the Figure 3.3 we have an architecture of how it is shaped where we can see that on the top of the pyramid we have the knowledge of the world **K**, in which the system is located. There are different layers and they are interconnected to make function as a feedback control system. Under the layer knowledge we have the layer of the objectives **O**. The interaction between these two layers are of the utmost importance since, if there are no objectives, the system will be in an indeterminate cycle. Below we have the states or situations **S**. It should be mentioned that depending on the objectives, the interaction of these stages will be the cause of the evolution of the system, converging to those objectives. And finally, in the lowest part of the pyramid are the actions **A**. These are the ones that will be in direct contact with the environment. To understand the operation of the architecture, we will begin by formally defining the four layers.

3.2.1. Situations, Objectives and Actions

Definition 5. *A world* $\boldsymbol{K}$ *is composed of objectives-*$\boldsymbol{O}$*, situations-*$\boldsymbol{S}$ *and actions-*$\boldsymbol{A}$*.*

$$\{S, O, A\} \subseteq K \mid S \subseteq \emptyset \ , O \subseteq \emptyset \ A \subseteq \emptyset$$

Firstly, the set of *situations* (S) contains information about parameters of an airplane (altimeter, airspeed, variometer...), environment.... On the other side, the set of *actions* (A) are what a pilot does physically (increase or decrease the engine power, turn the yoke to the left or right, ...). In this context, the ***situations*** and ***actions*** are represented by positive literals. For a certain situation, the challenge is to calculate the extensions that contain actions which allow to approach the desired ***objective*** (O). For instance, when an airplane is placed at the start point (S_p), Figure 1.9, assuming it has the authorization and it is possible to take-off, then the airplane take-off. This objective could be described by a default as follows:

$$\frac{(rest(x) \wedge authorization) : takeoff(x)}{takeoff(x)} \tag{3.1}$$

In the same way, we could describe when a plane (starts at some point a) wants to maintain an altitude greater than 1500 feet with a north direction, to reach to the point b. A default could be as follows:

$$\frac{((altitude(x) > 1500) \wedge compass(x, north)) : point(x, b)}{point(x, b)} \tag{3.2}$$

These are just two defaults as examples, but we can include many others in O. We consider two kind of objectives, short and long-term.

3.2.2. Short- and Long-term Objectives

When a pilot has a disturbance of any kind, he will naturally move away from the **objective** (O), this it could be land, take-off, climb... However, he must make actions to achieve the goal. A pilot is in constant revision of knowledge, taking information from the cockpit, environment and even from the control tower. Additionally, a pilot should respect air regulations and navigation laws. For a better understanding, we introduce the following concepts. The ***short-terms*** occur when there are perturbations and airplane moves away from the long-term objective. Thus, a pilot will find another short-goal to get closer and converge. For instance, when an airplane is climbing (from the point a to the point b, Figure 1.9) to 1500 feet and there are wind disturbances, Equation (3.1) is considered a sub-goal. On the other hand, a ***long-term*** objective is, for instance, maintain a steady flight for 5 minutes with an altitude of 1500 feet, Equation (3.2) is considered a long objective.

Definition 6. *In the world* **K**, *there is always a resilience trajectory* R.

$$\forall S, \forall O, \forall A \subseteq K \; \exists \; R \tag{3.3}$$

Short-term objectives have very fast change in comparison with long-term. Nevertheless, short-terms will allow to achieve long-term. As the system evolves and disturbances appear, exploration is an important stage of the model. Because this part it is the main process to find different sub-goals that will allow to absorb a shock ζ. Sub-goals g are related to the extensions since they contain actions to converge to the final goal. It is so that the system can jump between sub-goals and have a resilient behavior. So to apply the property of resilience in our model, inside the world K we will study the property of resilience, Figure 3.4. Since we know that the KOSA model has an evolution over time, we

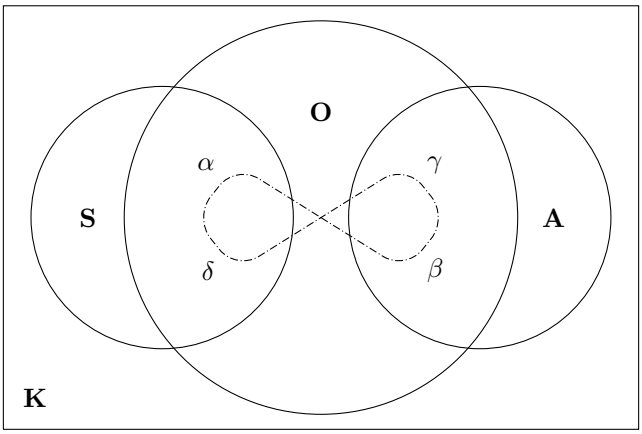


Figure 3.4: Representation of KOSA model.

are interested to study its form and properties. We can define a trajectory which has to satisfy 4 properties: ***reorganization*** (α), ***exploration*** (β), ***release*** (γ) and ***conservation*** (δ), Figure 3.4 (Crawford S Holling 2001; Sitterle, Freeman, S. R. Goerger, et al. 2015; Zhang, Mahadevan, Sankararaman, et al. 2018). According to the theory made by Holling, we can identify the four properties for our system. The first corresponds to the exploration, in this part is where we have a situation and we will look for or explore solutions. These solutions are the fixed points of a default theory, Section 2.3. Organization and conservation are contained in what we define as non-probabilistic decision making. It is important to note that the model used creates an operational space where the alternatives are sorted according to the losses and then choose the best and achieve a desired goal. Finally, we can define the release as the interaction of our system with the environment in which situations develop. In other words, when the system has chosen a solution, it will have to execute actions that will move towards an objective. To know if a goal was completed or to know the distance to it, we consider the Minsky's model.

3.2.3. Minsky's Model

In this thesis the control part is made thanks to Minsky's model. This model was created by Marvin Minsky, who was one of the fathers of artificial intelligence, along with others such as John McCarthy, Allen Newell and Herbert Simon. Marvin Minsky created a general concept that can be applied in both psychology and engineering. The principle of this model lies in the fact of having three fundamental parts. The first is a current situation in which an event develops, the second is a situation in which we want to be. And finally, we have a result of the difference between the current situation and the situations we want to be. This result will be the corresponding actions to reach the desired situation, Figure 3.5. Once the objective is achieved, we can have more future objectives, thus, repeating the same principle to converge with others (Minsky 1974; Minsky 2006).

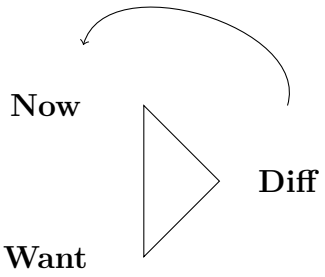


Figure 3.5: Minsky's model.

3.2.4. Continuous Model

This representation allows us to describe, as in the Figure 3.2, the solutions and internal connections for every situation S. Thanks to this, we can see how strong or weak is the model when perturbations are present. But if we would like to see the resilience we should increase one dimension. In K there is a set of situations, objectives and actions. We can find a dynamic of the fixed points, take up the meaning of Reiter (1980), remembering that the fixed points are solutions of a situation. So if these points have a dynamic, we can make a projection towards and study the trajectory or orbits of these points. That will happen when the system evolves from a situation to another future situation.

The representation in Figure 3.6 is proposed for a better understanding the property of resilience. This is why we have two sets: **K** and Υ. In Υ is contained the resilient behavior. The behavior can have different orbits. As Crawford S Holling (1973) established it may be an ∞ orbit. In the same manner, we can have circular or elliptical orbits. These forms will depend strongly on the connection of the internal states, the closer they are to each other, the system will

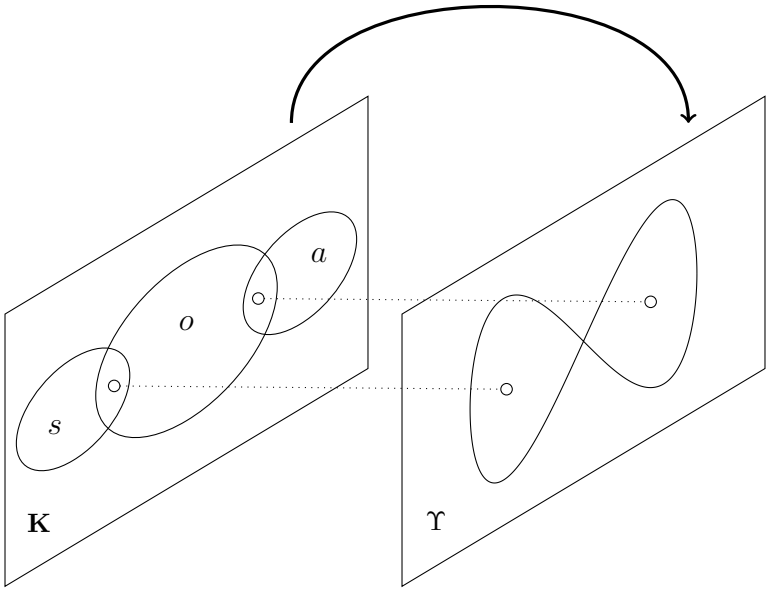


Figure 3.6: Trajectory of resilience.

respond quickly to disturbances. This is also related to the distance of the horizon between the objectives, while the shorter the horizon, the quicker it will react. Otherwise it will take more time to react if the horizon between objectives is long. In the world Υ there is a function that allows visualizing the dynamics of situations, objectives and actions. Mathematically it is defined as:

$$\Upsilon(f) : S \cap O \rightarrow O \cap A$$

In this plane only the interaction two states can be useful, for example of the Minsky's model these states can be the current and future situation. Being able to capture the behavior of the world K.

3.2.5. Discrete Model

We take a look at the discrete representation, Figure 3.7. At the begging S_p the computation of four extensions: $\{g_0, g_1, g_3, g_5\}$, according to our decision-making model g_1 is chosen and then the system interacts with the environment. At some point, disturb ζ_1 occurs and extensions are computed one more time: $\{g_1, g_4, g_5, g_6\}$, the better solution is g_6 and then interaction happens again. This process occurs every time disturbs ζ appear. In this sense, computing and choosing extensions, trajectories $(\triangle, \star)$ are created. For the first resilient trajectory $(\triangle)$, we have: $R_\triangle = \{g_5, \zeta_0, g_4, \zeta_1, g_3, \zeta_2, g_6, \zeta_3, g_5, \zeta_4, g_4, \zeta_5, g_6, \ldots\}$ and for the second trajectory $(\star)$, we have: $R_\star = \{g_1, \zeta_0, g_6, \zeta_1, g_3, \zeta_2, g_6, \zeta_3, g_5, \zeta_4, g_4, \zeta_5, g_1, \ldots\}$ This representation of trajectories was made with a small number of extensions

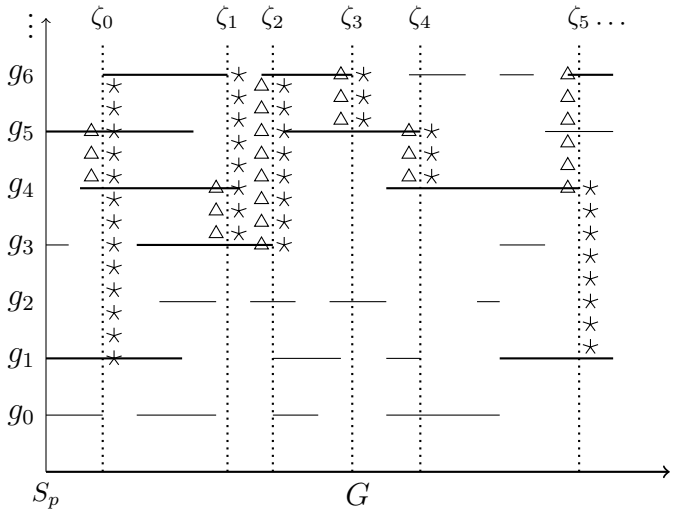


Figure 3.7: Evolution of a goal G, switching sub-goals g when a disturb ζ occurs.

and disturbances, for a greater number we have the following definition.

Definition 7. *The convergence of an objective G is the sum of the product of the sub-objectives g and disturbances ζ.*

Formally we can define the sequences of the trajectory as the sum of the product of the subgoals and shocks/disturbances:

$$\bigcup_{i=0}^{\infty} G_i = \bigcup_{i=0}^{\infty} g_i \cdot \zeta_i \tag{3.4}$$

3.3. Piloting as a Resilient System

In this section, we consider piloting as a resilient system. We must make use of some principles already explained. It is important to emphasize that to be realizable this example, we must take into account normal defaults, normal opportunistic function and Minsky's model. These three points are make possible to explain this ability of systems to absorb shocks and converge to a goal. An interesting point of the model is if we increase the number of defaults, we will increase the degrees of freedom. This is an important remark, because we consider degree of freedom a space in **O** where it could pass a trajectory. In this example, we describe a "take off" situation, where extensions are calculated, then choosing the best solution to converge to the objective. Supposing that all conditions are ready, and at some time t he will suffer a disturbance D. We can see in the Figure 3.9 others goals which are different flight states. We start in O, where a glider has not airspeed, $v \approx 0$, so it is considerate rest state (RS). If

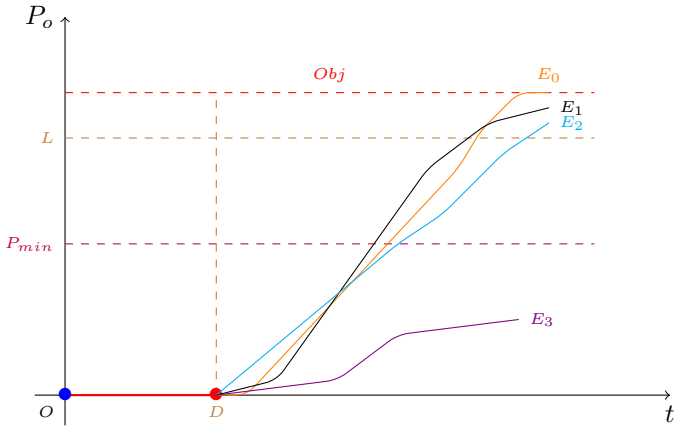


Figure 3.8: Takeoff behaviors.

we do not have the information about authorization, so we consider that authorization is false. We have a predicate "$glider(X)$", it is a formalization of "glider has X". Where $X = \{x_1, x_2, x_3, \ldots, x_n\}$ is a vector with all information from the instruments. Next predicate "$glider(airspeed\_zero)$" is a formalization of "glider has not airspeed"[1]. So, in RS has:

$$W = \{glider(airspeed\_zero), glider(altimeter\_zero), glider(variometer\_zero), glider(\neg motor\_on), glider(batt\_status\_green)\}$$

$$d_{10} = \frac{g(airs\_zero) \wedge g(var\_zero) \wedge g(\neg motor\_on) : pilot(yoke\_p\_neutral)}{pilot(yoke\_p\_neutral)}$$

$$d_{11} = \frac{g(airs\_zero) \wedge g(var\_zero) \wedge g(\neg motor\_on) : pilot(yoke\_r\_neutral))}{pilot(yoke\_r\_neutral)}$$

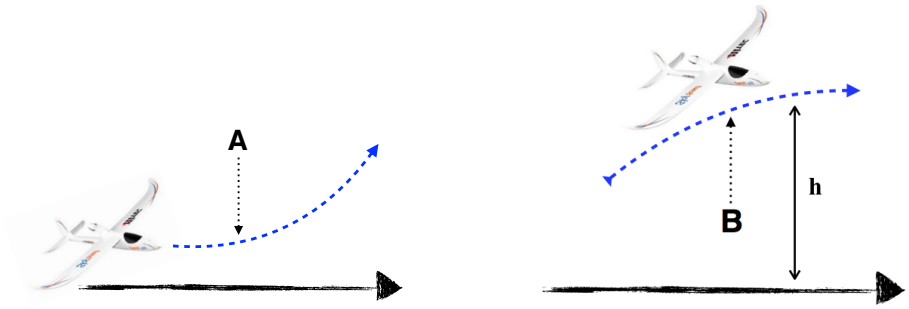
$$d_{25} = \frac{g(var\_zero) : \neg pilot(motor\_on)}{\neg pilot(motor\_on)}$$

$$E_0 = \{d_{10}, d_{11}, d_{25}\}$$

$$d_{11} = \frac{g(airs\_zero), g(var\_zero), g(\neg motor\_on) : pilot(yoke\_r\_neutral)}{pilot(yoke\_r\_neutral)}$$

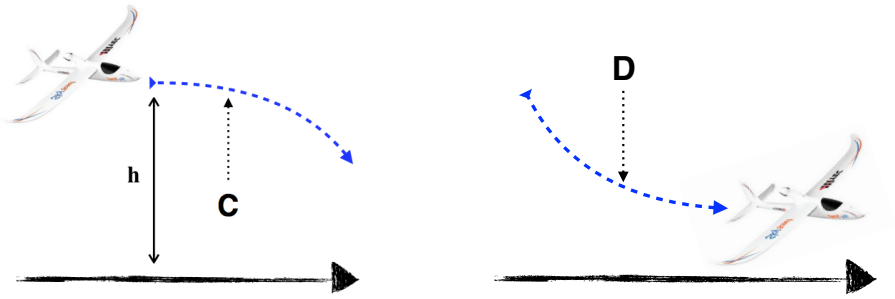
$$d_{25} = \frac{g(var\_zero) : \neg pilot(motor\_on)}{\neg pilot(motor\_on)}$$

[1] As a matter of space, we use the following syntax: glider=g.



(a) Objective 1

(b) Objective 2



(c) Objective 3

(d) Objective 4

Figure 3.9: Different situations in a flight.

$$d_{28} = \frac{g(var\_zero) : pilot(yoke\_push)}{pilot(yoke\_push)}$$

$$E_1 = \{d_{11}, d_{25}, d_{28}\}$$

For the moment, we control three variables: pitch, roll and motor. With pitch and roll, glider can move in all directions and motor allows to increase or decrease glider airspeed. Using our default, extensions are calculated. In E_0, a pilot will put: pitch and roll neutral position and motor off (d_{10}, d_{11}, d_{25}). On the other hand, in E_1, a pilot will put: positive pitch and roll neutral position and motor off (d_{11}, d_{25}, d_{28}). In both cases, glider speed is zero. At that point, if a pilot has not the authorization he can not take-off. Otherwise, he will take-off as a sub-objective, then will climb as another sub-objective, and so on until he reaches an objective.

We present a model that describes the evolution of a pilot reasoning. In reality a pilot makes two movements, he observes the horizon and next the cockpit, after that he does actions, he repeats this over and over again. This dynamic could be represented such as Figure 3.10. The model has transitions but the notion of

time is considered as an external parameter, that is, transitions only occur between situations. For example, if we have a situation s_i and if it is possible to go to the situation s_{i+1}, we should do actions. Firstly, we start with a default theory

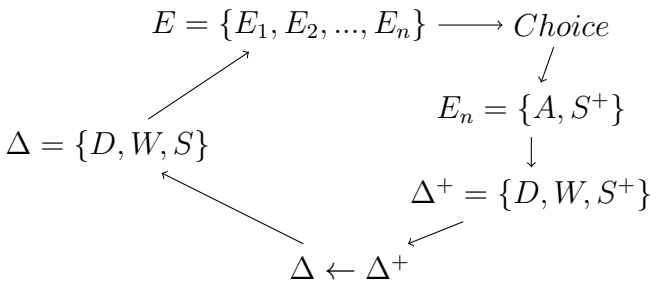


Figure 3.10: Resilient System of a pilot Based on Default Logic.

$\Delta = (D, W, S)$. Where D are the set of defaults, W are the set of FOL and S are the parameters of an airplane, environment, control tower... We are considering that Δ is a default theory before a transition and Δ^+ is a default theory after a transition. Similarly, S is a situation observed before a transition and S^+ is a situation observed after a transition. From Δ, the set of extensions E is computed. Each extension contains actions. Once we have the solutions we must choice the better extension that brings us closer to the goal, then decision-making is based as before. After a pilot applies actions he takes observations again (cockpit and environment) passing information from S to S^+. Then it goes back to Δ to compute extensions and choose the better one again. Sometimes for an airplane it is impossible to converge to the desired goal and alternative objectives must be found. In the set of objectives (O) the property of resilience is carried out. The following algorithm is the representation of the KOSA model, where it can be seen that the convergence to a target is made by an epsilon error. This error corresponds to the difference in the Minsky model. Once the difference is less than the epsilon error, this indicates that the objective has been met and another new objective is sought. In classical control, the study of dynamic systems, such as electrical, mechanical, hydraulic... can be modeled using block representation. These could have positive and/or negative feedback. If we consider two systems, **A** and **B**, which are connected. Positive feedback is a process in which the effects of small perturbations in **A** affect **B**, producing disturbances in **B** which will also affect **A**. This can lead to the collapse of both **A** and **B**. Positive feedback has to cause instability, generally the response is increasing exponential, increase of oscillations, chaotic behavior or divergence of the equilibrium point. Negative feedback is when a system reduces variations in its output, which are caused by changes in its input. Negative feedback leads to stability or equilibrium point, reducing the effects of disturbances. Resilience includes both feedbacks, having the ability to organize the behavior of a system allowing to use the same feedback connection such as both positive and negative. Thanks to the use of

Algorithm 2 Evolution of resilient model

Require: $\{D, W, S\} \notin \emptyset$
  procedure EVOLUTION(Δ, Δ^+)
    while $(O \notin \emptyset)$ **do**
      while $(||\vec{O_i}|| > \epsilon)$ **do**
        $E \leftarrow \Delta = (D, W, S)$
        $A \leftarrow Min\ Max\ E$ ▷ Choice
        return A
        $\Delta^+ = (D, W, S^+)$
        $S \leftarrow S^+$
      end while
    end while
    return A
  end procedure

a normal default theory, normal model of decision-making and Minsky's model, our proposition of a model that can capture the property of resilience, the interconnections of the states of the system, whether they have positive or negative feedback, it can be possible.

Conclusion

In this part was presented the definition of resilience, and the KOSA model. The KOSA model is a generalization for the representation of resilient systems. This model is composed of situations, objectives and actions. We show our complete model which is the model based on the non-monotonous logic, the decision making with the opportunistic criteria and the Minsky's model which describes how the mind get goals by changing the set of axioms in use (Minsky 1974; Minsky 2006). From our model in the Figure 3.10, we could consider the "Now" such as the actual situation S and "Want" such as S^+, the long-term objectives. The differences will be the actions that we should do to converge to the main objective. Thanks to these three great notions we can capture the property of resilience.

The control of our model is done thanks by both concepts, the property of resilience and Minksy's model. The principle of Minsky's model lies in the fact of having three fundamental parts. The first is the current situation in which an event develops, the second is a situation in which we want to be or arrive. And finally, the result of the difference between two situations. This result will be the corresponding actions to reach the desired situation. Once the objective is achieved we can have more future objectives. The KOSA model converges to sub-goals and goals when the current situation is inside of an epsilon error. This error corresponds to the difference in the Minsky model.

4 Practical Case

The objective of this section is to describe the system which we validated our NMR and resilient model. Computer on board, sensors and electrical elements are described. SWI-Prolog setup for our embedded computer and Microcontroller Unit (MCU) interface are described. Eventually, solar cells calculation about current and voltage needed are presented. The objective of this section is to show practical results, including articles published about this research.

Summary

4.1. Motor-glider

Actually new directions open to use airplane or any aerial vehicle for different applications in an autonomous way. In this thesis we are using an UAV, but it is not confined to the military fields. They are used in many areas such as agriculture, construction, entertainment. . . A glider is kind of airplane which has really good advantages in terms of weight and aerodynamics. The principle of a glider is that it uses the “vertical” wind (thermal and dynamic energy). As in the nature, there are various species of birds using the same principle: albatrosses and condors. An experienced pilot of glider can maintain a flight for a long time, in a day with good weather, without using the engine as propulsion.

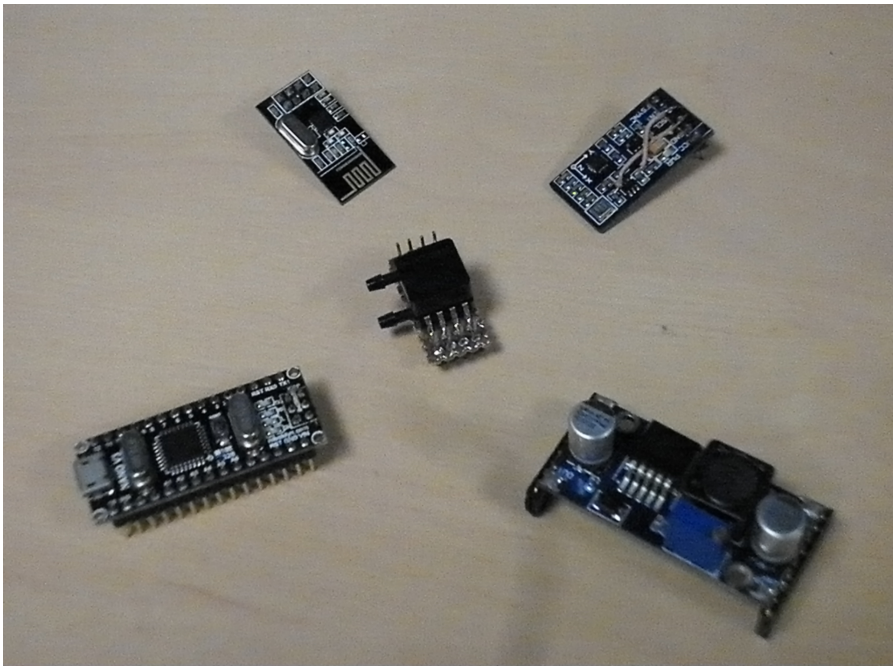
We are using a motor-glider which is a glider equipped with an engine, a scale model. In fact, it can be launched with just one hand. In a real-scale glider it must be guided by a cable with a master airplane which will reach a certain

altitude. For a real motor-glider this can take-off and climb without assistance, in contrast with a “normal” glider that is non-motorized. The problems of piloting a motor-glider are the same as piloting an airplane.

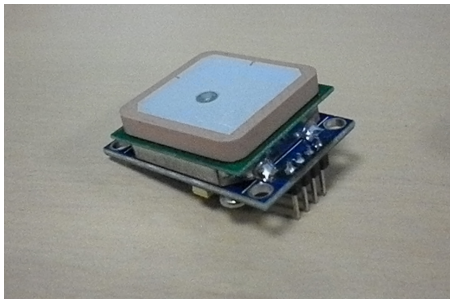
We are currently developing a prototype which will allow us to validate our model. The airplane that we use is of the registered trademark, SkyScout R2GO Hitec. This included different accessories, Figure 4.1, such as remote control, Lithium-Polymer (LiPo) 1300mAh batteries, Hitex HS-55 servo-motors, C2812-1100 Brushless motor, Minima 6S receiver, Brushless Motor Controller (BMC) 18 amp...

Figure 4.1: Motor-glider used.

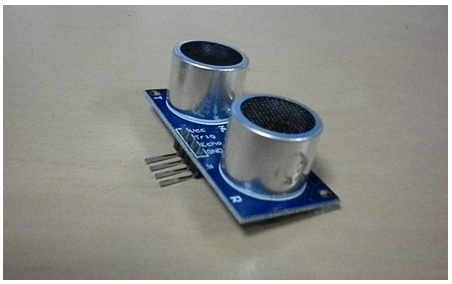
The model has the next specification: wingspan: 1366 mm, weight:(all-up weight min.) 700 grams (g), length: 977 mm, wing area: 28 dm^2 (wing + tail plane, excl. fuselage) and wing loading: 25 g/dm^2. Electrical system is composed by a 11.1 Volts (V) LiPo 1300mAh battery. This is the main source of energy for the receiver, servo-motors and Brushless motor. There is a voltage regulator between the LiPo battery and MCU receiver which brings 5 V, for the servo-motors and MCU.



(a) Up to down: Wireless,IMU, Pitot sensors, MCU and buck converter.



(b) GPS sensor.

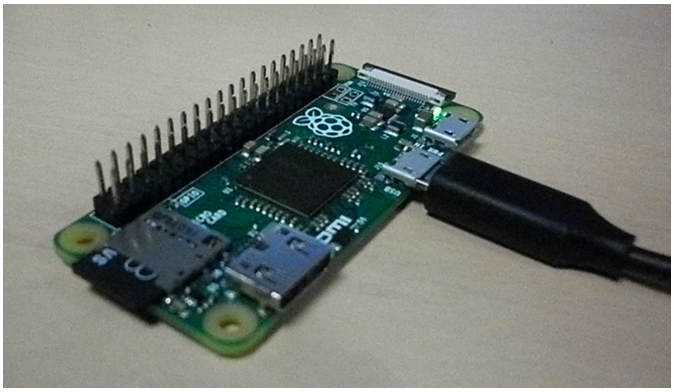


(c) Ultrasonic sensor.

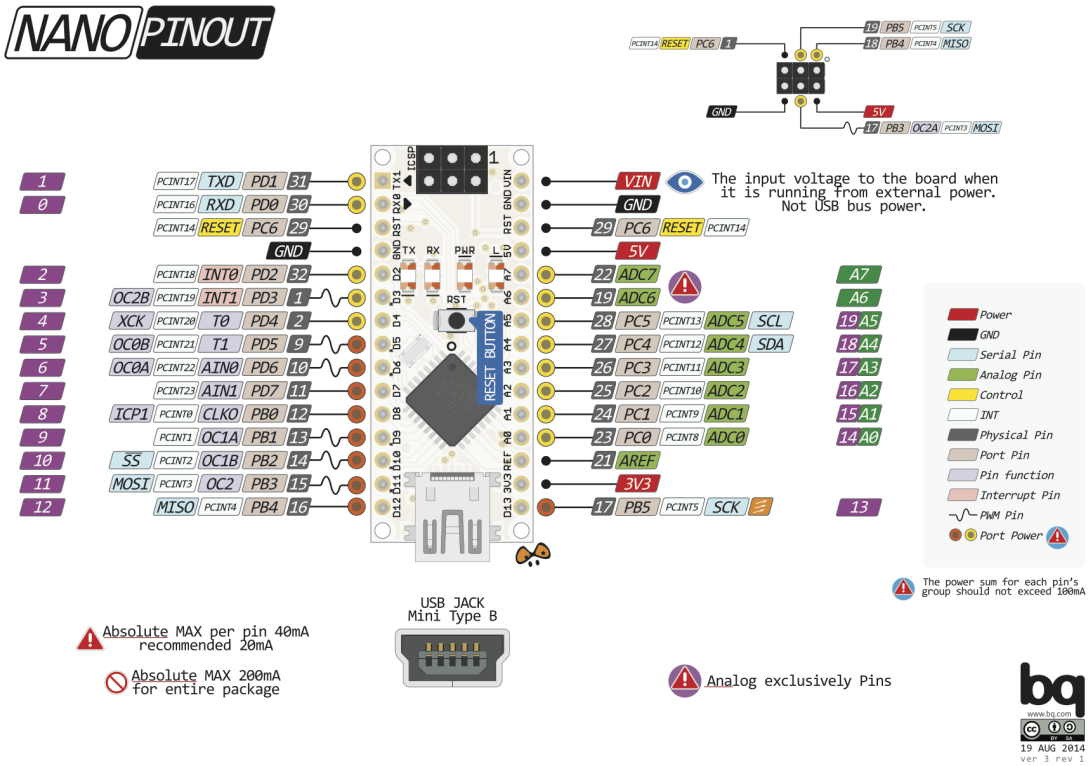
Figure 4.2: Devices used.

4.2. Embedded Computer

In the market there are a lot of choices of embedded computers, one of the constraints is about consummation of energy, Table 4.1. We choose a microcomputer called *Raspberry Pi Zero* which has 0.8 Watts (W) of consummation. Within the range of this type of microcomputers this is the most optimal version in energy. The dimensions of embedded computer are 65mm long x 30mm wide x 5mm thick, it is thin because of miniaturization of components and connections. The embedded computer is powered with 5 volts and is based on an ARM architecture. It is equipped with different serial communication protocols (Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C), Universal



(a) Microcomputer



(b) Interface MCU pinouts

Figure 4.3: Embedded systems.

Asynchronous Receiver-Transmitter (UART)). It includes 40 digital input/output pins, some of them are destined to 5 V, 3.3 V and ground, 0 V. The computer has a micro-SD reader, where the operating system could be executed. A micro-HDMI input where a screen can be connected. Two connections micro-usb, one is to connect peripherals (keyboard and mouse) and the other is for the power supply. We can connect different sensors to the microcomputer. For example,

inertial sensor like a gyroscope which measures the rotation of the motor-glider. It is very accurate for fast movements, that is, fast rotations. An accelerometer that can measure static and dynamic forces at any given time. And eventually, a magnetometer which allows us to estimate the direction of the motor-glider by detecting the magnetic flux on earth. With these three sensors we can have almost all the information of cockpit.

Embedded Systems	RAM	CPU	Power
Model B Rev.2	512MB	700MHz ARMv6-rev 7	3.5W
Model Zero	512MB	1GHz Single Core	0.8W

Table 4.1: Embedded computer CPU table.

How control a servo-motor?

This is a digital modulation, called PWM, which can be used to control the position and speed of motors and servo-motors, but also the intensity of the light in a led... This type of digital signal is characterized by having high (+ 5V) and low (0V or earth) values for any time. We can change the amount of time in which this signal remains high or low, this is called *Duty Cycle*. This parameter is dimensionless and is usually a percentage, that is, 100% corresponds to + 5V (high) and 0% corresponds to 0V (low). The microcomputer has a problem with the generation of this type of signals, that is, it is very unstable. For this reason we use an interface MCU that provides a better PWM signal in terms of stability. This is, we connected a LED to the embedded computer and we generated a digital signal with a frequency fixed. The result obtained was a variable light intensity. If we do the same experiment with our MCU, the result is more stable. We show the most important characteristics of two models of embedded computer. In Figure 4.4 we can see the variation of the duty cycle generated from the MCU interface and the angle of a servomotor. This curve was obtained with a linear function that takes the value of the duty cycle.

4.2.1. Setup Access

First we need to download and install latest Jessie. We're using Jessie Lite but plain Jessie Raspbian should work too. You need May 2016 or later (tested with 2016-05-27). Then edit `config.txt` and also `cmdline.txt`. After burning the SD card, do not eject it from your computer! Use a text editor to open up the `config.txt` file that is in the SD card post-burn. Go to the bottom and add `dtoverlay=dwc2` as the last line. Save the `config.txt` file as plain text and then open up `cmdline.txt`. After **rootwait** (the last word on the first line) add a space

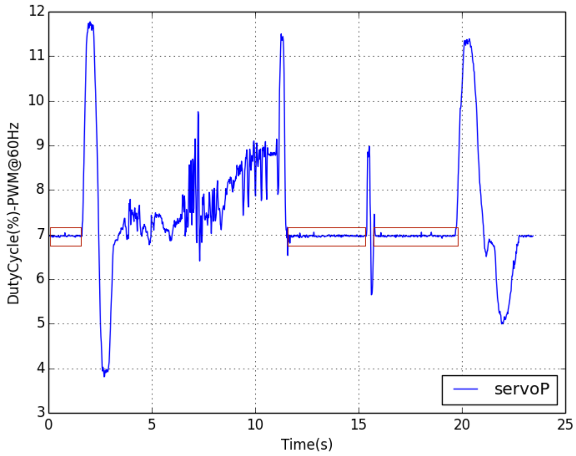


Figure 4.4: Controlling a servo with PWM.

and then **modules-load=dwc2,g_ether**. Put the micro sd card on the raspberry, and connect only the micro usb to the usb computer.

If you enable SSH on your Pi, you can then also SSH in to "raspberrypi.local".

```
$ ssh pi@raspberrypi.local
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: POSSIBLE DNS SPOOFING DETECTED!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The ECDSA host key for raspberrypi.local has changed,
and the key for the corresponding IP address fe80::ac0a:1bba:19b7
    :9412%bridge100
is unknown. This could either mean that
DNS SPOOFING is happening or the IP address for the host
and its host key have changed at the same time.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle
    attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:tF+CT6tc8I+EO/FVTaTjPXuTqusHAgs2fJMFz4wQl0g.
Please contact your system administrator.
Add correct host key in /Users/joseluis/.ssh/known_hosts to get rid
    of this message.
Offending ECDSA key in /Users/joseluis/.ssh/known_hosts:1
ECDSA host key for raspberrypi.local has changed and you have
    requested strict checking.
Host key verification failed.
```

Here the solution, edit the $\sim/.ssh/known\_hosts$ file and delete the line that

contains the offering key: "**fe80**: : **ac0a**: **1bba**: **19b7**: **9412**%**bridge100**"

The embedded system that is used is based on an ARM architecture. It has different serial communication protocols. It can be fed with 5 volts. It is made up of 40 digital input and output pins, some of them are destined to 5v, 3.3v and ground. With these voltages different sensors can be fed. A micro-sd reader, where the operating system could be executed, in our case it is one based on debian. A micro-HDMI input where a screen can be connected. Two connections micro-usb, one is to connect peripherals (keyboard and mouse) and the other is for the power supply. finally, it has a connection for a camera.

To connect to the embedded computer, you simply plug a micro-USB cable to a USB port on your computer. In this way, we will have the power supply and communication for the embedded computer. Previously, a configuration was made to emulate Ethernet via USB connection. Since your computer you can execute the following command from a terminal.

```
$ ssh pi@raspberrypi.local
```

Following these instructions, we can install SWI-Prolog on the embedded computer[1]:

```
$ git clone https://github.com/SWI-Prolog/swipl-devel.git
$ cd swipl-devel
$ ./prepare
$ ./configure
$ make
$ [sudo] make install
```

In this last part you must be patient because it may take a few minutes. But if you did all the previous steps without any error message, you can run SWI-Prolog from the embedded computer by executing on terminal: `swipl`. To know the version: `swipl -version`. In our case, we are using: `SWI-Prolog version 7.7.8`. To execute any command you must include a dot at the end, then press enter.

4.2.2. Running a Situation

We define clauses as rules of piloting. These rules follow a specific syntax which has the form: $cl(\texttt{text}, \delta, A, C, \omega)$. Where `text` could be a chain of characters as comment, δ could be a real fact "`hard`" or default "`def`", A and C are the prerequisite and consequent, and finally ω is the priority weight, (A. Doncescu and P. Siegel 2015). To find the solutions it is necessary to solve a set of logical equations. Computational difficulties are the biggest challenge. However, using Horn clauses, complexity is reduced and the expressiveness suffices. This kind of negative clause is used to express events that are not possible (mutual

[1] If we have a problem about "autoconf". We should do: `sudo apt-get update` and then `sudo apt-get install autoconf`

exclusion), for example, physically a pilot can not have the rudder up and down at the same time. The theoretical computational complexity is $\sum_2^p$. In our implementation normal defaults ($\frac{A:C}{C}$) and Horn clauses are used only. Allowing to compute the extensions in a quasi-linear time. However, if we increase the number of defaults, the calculation time does not increase exponentially.

Being in the right directory you can run a program ".pl" with "`consult(file_name).`", If the compilation was done correctly, a message will appear: `true`. In our case, we have: "`consult(condor_inference).`" and then we use a clause: "`run_condor2.`", which will compile two parts. The first one is the non-monotonic reasoning part and the second one is the representation of the current situation under a syntax described before. The informations captured by sensors correspond to real facts and they can be described as follows:

```
cl('',dur,[], glider(airspeed_low),[]).
cl('',dur,[], glider(pitch_stable),[]).
cl('',dur,[], glider(roll_stable),[]).
cl('',dur,[], glider(altimeter_low),[]).
cl('',dur,[], glider(variometer_zero),[]).
```

This facts describe the current situation of an airplane, in our case a glider, having no inclination or rotation, low airspeed and altitude, and no vertical speed. With this we can assume that the glider is motionless. Due to limited space to write, we will take the following notation; for the glider g, zero vertical speed is var_zero, no inclination or rotation is pch_stb, rll_stb, and authorization is $auth$. In the same way defaults are represented as:

```
cl('',def,[g(var_zero),g(pch_stb),g(rll_stb),auth],pilot(motor),[]).
```

This default can represent a possible solution, where the prerequisite (information of six-pack and authorization) is true and there are not contradictions with the conclusion, then we jump to $pilot(motor)$. This conclusion is an action of turning on the engine to take off. There are 12 possibles actions: 3 for the pitch, roll, yaw and motor. If we consider the following information, $G : \{g(pitch\_stable), g(roll\_stable), g(motor\_off), g(low\_alt), g(low\_airspeed)\}$, we ask to prolog what we can do with, could we take-off? Using the syntax described above we can use G as real facts. Then we consult our code and we obtain 5 different extensions. Table 4.2 contains the solutions calculated: E_0 has $\{d_{16}, d_{17}, d_{20}\}$. In the same way E_1 has $\{d_{16}, d_{17}, d_{21}\}$, likewise for the others defaults. We can use the following notation, $\frac{A:C}{C} \equiv A : C \rightarrow C$. Defaults used

are:

$$d_{16} = g(roll\_stable) : pilot(yoke\_roll\_neutral) \rightarrow pilot(yoke\_roll\_neutral),$$
$$d_{17} = g(pitch\_stable) : pilot(yoke\_pull) \rightarrow pilot(yoke\_pull),$$
$$d_{18} = g(low\_alt) : pilot(yoke\_roll\_neutral) \rightarrow pilot(yoke\_roll\_neutral),$$
$$d_{19} = g(low\_alt) : pilot(yoke\_push) \rightarrow pilot(yoke\_push),$$
$$d_{20} = g(low\_alt) : pilot(motor) \rightarrow pilot(motor),$$
$$d_{21} = g(low\_alt) : \neg pilot(motor) \rightarrow \neg pilot(motor)$$

From G the best extension we can choose is E_3. Because it has the combinations

Extensions	d_{16}	d_{17}	d_{18}	d_{19}	d_{20}	d_{21}
E_0	✓	✓			✓	
E_1	✓	✓				✓
E_2	✓			✓		✓
E_3		✓	✓		✓	
E_4		✓	✓			✓

Table 4.2: Extensions calculated for a takeoff situation.

of the actions $\{pilot(yoke\_pull),\ pilot(yoke\_roll\_neutral), pilot(motor)\}$ needed to reach the goal: **take-off**, assuming that the authorization is true. Since the extension E_0:

$$\{pilot(yoke\_roll\_neutral), pilot(yoke\_pitch\_neutral), pilot(motor)\}$$

The result is a straight flight that's not the goal, it would probably be a solution to be able to have more speed and later take-off. The extension E_1 has: $\{pilot(yoke\_roll\_neutral),\ pilot(yoke\_pitch\_neutral), \neg pilot(motor)\}$, there is no movement on the glider because the engine is off. The E_2 extension has: $\{pilot(yoke\_roll\_neutral), pilot(yoke\_push), \neg pilot(motor)\}$, same result as the previous extension, no movement because the engine is off. And finally, the E_4 extension has: $\{pilot(yoke\_pull), pilot(yoke\_pitch\_neutral), \neg pilot(motor)\}$, same result as the previous extension, no movement because the engine is off. After making a decision and applying actions, the model will capture information from the sensors to close the cycle, Figure 3.10. This will lead to change objectives, for example after taking off now the new one will be to climb a certain height, if possible. In Figure 3.9, we can see different situations of an airplane, but also different objectives, (J.-L. V. Medina, Pierre Siegel, and Andrei Doncescu 2017).

In Table 4.3, we can make an analysis when our system has enough information it can reduce the CPU calculations, 7 facts with 95% of CPU reducing logical inferences per second (Lips), therefore more solutions (13 extensions). On the

Facts	Extensions	Instanced clauses	CPU	Lips
7	13	115	95%	114,131
5	13	113	98%	117,176
4	10	112	97%	130,098

Table 4.3: Comparative table on the results obtained from three different situations.

other hand, if it has uncertain and incomplete informations the interpreter will have to do more calculations, 4 facts with 97% of CPU increasing Lips, which is normal since it should demonstrate more clauses, therefore less solutions (10 extensions).

4.3. Sensors

We incorporated some sensor to bring autonomy by itself. We start with an inertial sensor which is an embedded system. This is in a single board there are three devices such as accelerometer, gyroscope, magnetometer and pressure sensor. Today we will analyze graphs with the accelerometer and gyroscope data to establish a semantic / syntax (if not the appropriate word) and be able to represent knowledge. This definition of semantics / syntax will allow us to use the algorithm (written in prolog) to make decisions (stabilize the plane, choose a trajectory, optimize energy, etc.) Pending, analyze the rotation files to establish the syntax of knowledge representation and then merge it with the prediction algorithm in prolog. To have a reference in the axes of the IMU with respect to the axes of rotation of the aircraft (glider) it is necessary that they are coordinated to carry out the representation of the rotation (in logic of first order).

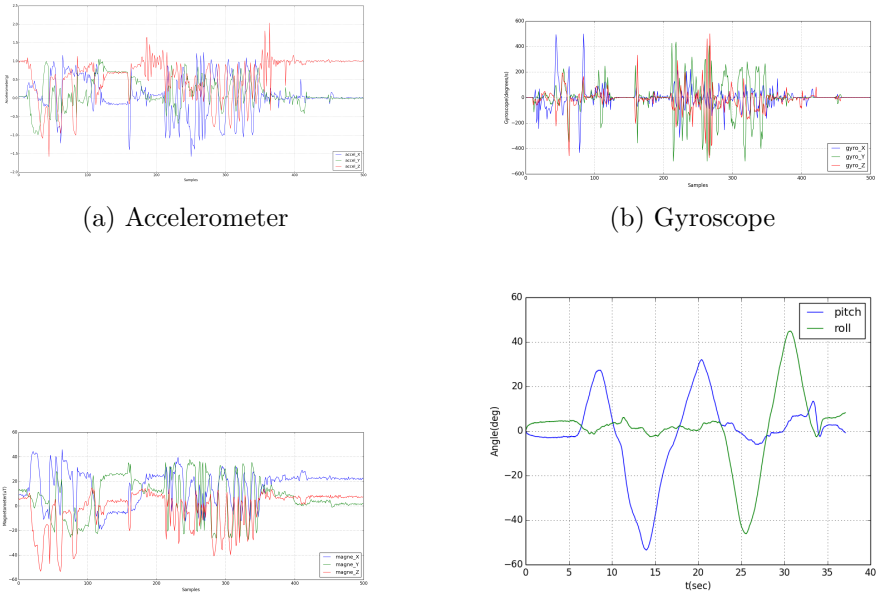
4.3.1. Inertial

The sensor that is being used is the MPU9255 of the WaveShare brand. It is a 10 DOF IMU sensor. This integrated by a gyroscope (detection in 3 axes), accelerometer (detection in 3 axes), electronic compass (compass-detection in 3 axes), pressure sensor and temperature. Which uses a serial communication protocol, I^2C. This type of sensor has the advantage of working with 5 V or 3.3 V. Thanks to I^2C embedded computer and IMU can communicate. Physical pins 3 and 5 (or GPIO 2-3) of microcomputer are destined for this purpose. Doing the right connections, we can execute from the terminal: *sudo python main_-MPU9255.py* and read data from the sensor. In case it does not work, verify the connections previously mentioned. A complementary filter, this is a filter that uses the data of the accelerometer and the gyroscope to have an accurate

signal of the movements made by the object. It is a filter commonly used for its efficiency and simplicity, Equation (4.1). Since it does not require complex operations and its calculation time is extremely short. Its implementation is very fast and easy.

$$angle_i = 0.98 * (angle_{i-1} + gyro * dt) + 0.02 * (acc) \tag{4.1}$$

The following graph shows the rotations of the X axis. We can see 3 different types of functions. One of them is the rotation using the accelerometer, the other is using the gyroscope; Both rotations are with respect to the same axis X. And the last graph is the fusion of the previous information, that is to say, using the information of accelerometer and of the gyroscope a better estimate of the real angle is carried out, Figure 4.5. Graphic representation in space: pitch, roll and



(a) Accelerometer (b) Gyroscope

(c) Magnetometer (d) Sensor fusion

Figure 4.5: Data from IMU.

yaw. It is used a complementary filter. The gyroscope measures the rotation of the object. It is very accurate for fast movements, that is, fast rotations. But a disadvantage is that over time has a cumulative error. The accelerometer on the other hand can measure static and dynamic forces at any given time. However, it captures data with a lot of fluctuating information (noise) and to have a good reading one must wait a considerable time, Figure 4.6.

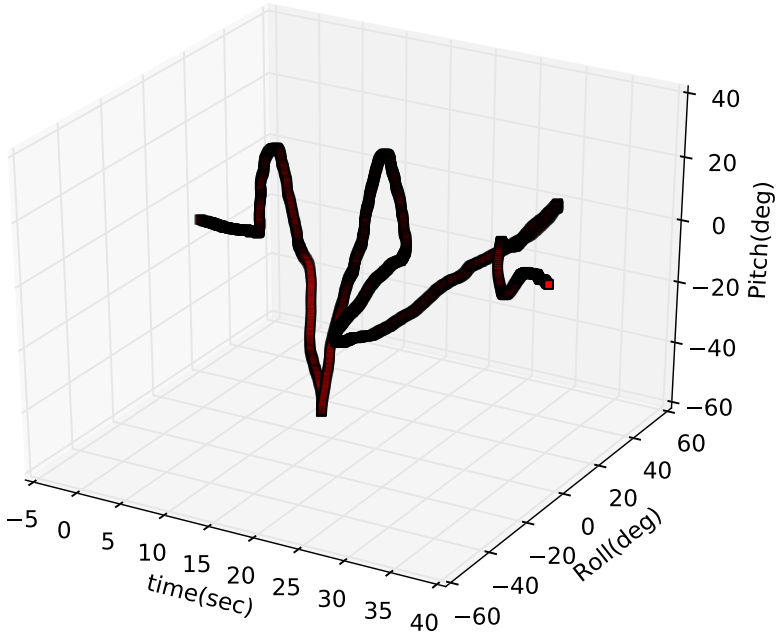


Figure 4.6: 3D representation of the pitch and roll.

4.3.2. Pitot Tube

Wind speed sensor plays an important role. This element is used to determine if the angle of attack should be increased or decreased so as not to lose lift. MPXV7002DP is a differential pressure sensor which allows to calculate the airspeed. The difference of pressures are analog values so these should be converted to digital, thanks to Analog to Digital Converter (ADC) and then a linear equation. To calculate velocity through the pressures, total pressure = static pressure + dynamic. Bernoulli's equation: $(p_s + \frac{r \times v^2}{2}) = p_t$, where r = density, v= velocity, p = pressure. So, $v^2 = \frac{2(p_t - p_s)}{r}$.

4.3.3. GPS

A GPS sensor used is a GY-NEO6MV2 New NEO-6M GPS Module NEO6MV2 with Flight Control EEPROM MWC APM2.5 Large Antenna. With the use of GPS you can locate an object on the ground, the principle is based on the triangulation of satellites that are in orbit. To access the received data, a communication protocol called NMEA must be configured, as well as it must be supported by the microcomputer. This is a sample of the type of data sent by the sensor.

```
$GPGGA,,,,,,0,00,25.5,,,,,,*7A
$GPGLL,,,,,,V,N*64
$GPGSA,A,1,,,,,,,,,,,,,25.5,25.5,25.5*02
```

```
$GPGSV,1,1,00*79
$GPRMC,,V,,,,,,,,,,N*53
$GPVTG,,,,,,,,,N*30
$GPZDA,,,,,,*48
$GPTXT,01,01,01,ANTENNA OK*35
```

Thanks to the GPS data it is possible to correct the wind drift, knowing previously the desired trajectory. As presented in the Section 1.2.

4.4. IMU Interfacing

Servo-motors, IMU,GPS, pitot tube... are connected a MCU. It is a Nano V3 ATmega328/CH340G. Compatible for Arduino Nano V3.0. It has built-in digital inputs and outputs, as well as the management of PWM signals. Supports serial communication protocols, which are ideal for the interconnection to other devices. It was not easy programming MCU from the embedded computer, via ICSP. Some connections are not directly linked. Signature problem and GPIO lines were the problem. Installing avrdude[2] tool, then adding following lines to "avrdude_gpio.conf":

```
# Linux GPIO configuration for avrdude.
# Change the lines below to the GPIO pins connected to the AVR.
programmer
id    = "pi_1";
desc  = "Use the Linux sysfs interface to bitbang GPIO lines";
type  = "linuxgpio";
reset = 8;
sck   = 11;
mosi  = 10;
miso  = 9;
;
```

This command verify communication and signature:

```
$ sudo avrdude -p m328p -C ~/avrdude_gpio.config -c pi_1 -v
```

Using the next connections[3], problems were solved. First, we make our code using "program_name.ino" (Arduino code). We compile, using "make" command, creating "build" folder. Inside the folder we get our "program_name.hex" then we need to write our program. This command burn the flash with a specific "hex" file (arduzero.hex).

```
$ sudo avrdude -p m328p -C ~/avrdude_gpio.config -c pi_1 -v -U flash:
   w:arduzero.hex:i
```

[2]https://learn.adafruit.com/program-an-avr-or-arduino-using-raspberry-pi-gpio-pins/overview
[3]https://www.arduino.cc/en/Tutorial/ArduinoISP

Programming from the Embedded Computer

We should do a previous configuration before burn a code from our embedded computer to the MCU interface. First, we should install avrdude compiler and define physical pins to establish a good In-Circuit Serial Programming –or *In-System Programming*– (ICSP) communication[4]. Next, another configuration[5] it should do, when Makefile is created it should add the next lines:

```
AVRDUDE_CONF = /home/pi/avrdude_gpio.conf
AVRDUDE_ARD_PROGRAMMER = pi_1
```

A file will contain the following lines;

```
ARDUINO_DIR = /usr/share/arduino
BOARD_TAG = uno
ARDUINO_PORT = /dev/ttyACM0
ARDUINO_LIBS =
#USER_LIB_PATH = /home/pi/SBR/arduino/libs
USER_LIB_PATH = /usr/share/arduino/libraries

AVRDUDE_CONF = /home/pi/avrdude_gpio.conf
AVRDUDE_ARD_PROGRAMMER = pi_1
```

If there is a problem, the address where this file is located should be: include /usr/share/arduino/Arduino.mk To carry out a test, an example code will be compiled from the microcomputer to later program that code in the MCU interface, file (.ino) is an example of:

```
void setup() {
// initialize digital pin LED_BUILTIN as an output.
pinMode(12, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(12, HIGH);   // turn the LED on (HIGH is the voltage
   level)
delay(100);                       // wait for a second
digitalWrite(12, LOW);    // turn the LED off by making the voltage
   LOW
delay(1000);                       // wait for a second
}
```

In the same folder it should contain both files: name.ino and Makefile. To compilate, we should execute “make” and with “make upload” we burn the code to the interface. This command is to know some information about our MCU connected.

```
$ sudo avrdude -p atmega328p -C ~/avrdude_gpio.conf -c pi_1 -v
```

[4]https://learn.adafruit.com/program-an-avr-or-arduino-using-raspberry-pi-gpio-pins/overview
[5]http://www.robot-maker.com/forum/tutorials/article/88-compilation-sur-rpi-et-upload-sur-arduino-directement-depuis-le-pi-en-ligne-de-commande/

Power System

The following diagram shows the complete diagram of the power system of the entire circuit. The solar stage is not connected, as well as the converter that recharges the 11.1 v battery. The part that includes the battery, regulator, microcomputer, interface, sensors and motors, is currently installed, Figure 4.7. The microcomputer, interface and different sensors are connected to their respective power supplies and communication protocols. Servomotors are also connected to the interface to control the angle.

Conclusion

Results of the different sensors are shown, the installation of the microcomputer is also described. The inertial sensor provides the accelerations, angular velocities and measurements of the earth's magnetic field, using this data and a complementary filter, we get the facts W. These three information, it allows us to know the orientation of the motor-glider in space, for instance, if it is going up, down, turning... Altitude is provided by the GPS module but also it is calculated by an atmospheric pressure sensor. Pitot tube is an instrument that allows to measure the static and dynamic pressure, and thus to know the airspeed of the airplane, based on the Bernoulli's equation. For obstacle detection an ultrasonic sensor is used, with a max. detecting distance of 4-5m. The aileron control is done by servomotors through PWM signal. The circuit on board is supplied with 11.1 V and 1300mAh LiPo battery. In the microcomputer, SWI-Prolog was installed. Until now, we have 110 defaults and the extensions are calculated in the order of milliseconds (Le, A. Doncescu, and P. Siegel 2013; A. Doncescu and P. Siegel 2015). We successfully installed "`SWI-prolog version 7.7.8`" in an embedded computer and a model of a non-monotonic reasoning was implemented. We tackled the problem of incomplete and uncertain information by formalizing the flight rules using default logic. We had good results in terms of calculation time, thanks to use Horn clauses and normal defaults only, since one of the restrictions is the low energy consumption (0.8 Watts), running at 1 GHz ARM11 (single core) and 512 Mb of RAM.

Conclusion

In this thesis it was possible to demonstrate that a problem can be modeled piloting rules in logic. Using FOL which is very expressive, and we can represent almost every situation. The limitation of the principle of non-contradiction of classical logic was overcome thanks to no-monotonic logic. Where default logic is a very effective tool to deal with issues such as exceptions, impartial and contradictory information. A default theory is a pair $\Delta = (D, W)$, where D is a set of defaults and W is a set of formulae in FOL. A default d is: $\frac{A(X):B(X)}{C(X)}$, where $A(X), B(X), C(X)$ are well-formed formulae. Intuitively a default means,"if $A(X)$ is true, and there is no evidence that $B(X)$ might be false, then $C(X)$ can be true". Once defaults, extensions are calculated and there may be no, one or several extensions. Normal defaults can guarantee at least one extension. We saw different decision-making models. We used an opportunistic model which creates an operational space where it shows us the losses that we should have before the event happens, then chooses the best option. In this manner, we optimised then a decision is made, referring to an "a posteriori" decision making. The biggest advantage of this model is that it has more flexibility in terms of predicting events, precisely because of the calculation of a regret matrix. The fact of choosing an extension under the help of a minimization or maximization function does not completely ensure the desired behavior. This was resolved by applying the property of resilience and Minsky's model. In the theory formulated by Crawford S Holling (2001), resilience is defined such as adaptive cycles. These are three properties: potential, connectedness and adaptability. Potential determines future solutions, it also determines the limits of what is achievable. Connectedness is the degree of connectivity between internal variables and processes, this is a measure that can describe the level of flexibility or rigidity of the control (sensitivity or not to disturbances). And finally, the ability to adapt, this is the resilience of the system, reflecting its vulnerability to an unpredictable or unexpected event.

We proposed a model called KOSA, which generalized the representation of a resilient system. This model is composed of a NML based on DL, non-probabilistic decision making and the Minsky's model. Thanks to this, we can determine a horizon of convergence. This horizon can be short or long according to the current situation and goal. In the same way, this model allowed us to know the behavior. A general model that can be used not only for piloting an airplane. But also in all system that human reasoning is involved. From solving problems like playing chess to driving a car. Because both in chess and driving a car, decision-

making under uncertainty is always present. A control part was made thanks to Minsky's model. In principle lies three fundamental parts; the first is a current situation in which an event develops, the second is a situation in which we want to be, and finally, we have a result of the difference between the current situation and the situations we want. This result will be the corresponding actions to reach the desired situation.

The airplane that we use is of the registered trademark, SkyScout R2GO Hitec. This included different accessories, Figure 4.1, such as remote control, LiPo 1300mAh batteries, Hitex HS-55 servo-motors, C2812-1100 Brushless motor, Minima 6S receiver, BMC 18 amp... We choose a microcomputer called *Raspberry Pi Zero* which has 0.8 W of consummation. Within the range of this type of microcomputers this is the most optimal version in energy. The dimensions of embedded computer are 65mm long x 30mm wide x 5mm thick, it is thin because of miniaturization of components and connections. Sensors were connected, the MPU9255 of the WaveShare brand. It is a 10 DOF IMU sensor. This integrated by a gyroscope (detection in 3 axes), accelerometer (detection in 3 axes), electronic compass (compass-detection in 3 axes), pressure sensor and temperature. Which uses a serial communication protocol, I^2C. This type of sensor has the advantage of working with 5 V or 3.3 V. A complementary filter was implemented, this is a filter that uses the data of the accelerometer and the gyroscope to have an accurate signal of the movements made by the object. It is a filter commonly used for its efficiency and simplicity. Since it does not require complex operations and its calculation time is extremely short. Its implementation is very fast and easy. Finally, we successfully installed "`SWI-prolog version 7.7.8`" in an embedded computer and a model of a non-monotonic reasoning was implemented.

Future Works

The implementation is currently in progress with good results, sensors were connected and Prolog was installed. This is a motivation to have a resilient model able to find thermal and to be able to fly as long as possible autonomously.

4.5. Energy System

This is currently taking place, but we are trying use a solar cells system with the next characteristics: Polycrystalline Solar Photovoltaic Cell, this one can provide 0.5 V, 0.2 W and 52 $\times$ 26mm as physical dimensions. Because LiPo batteries need a properly charging and discharging of voltage and current, we should use a control charger: 3S, 10A, 12 V Lithium Batterie Charger Protection. Solar cells system provided around +12 V, but embedded computer works with +5 V so we should include also a Power Control System: XL6009 DC-DC Adjustable Step-up boost Power Converter.

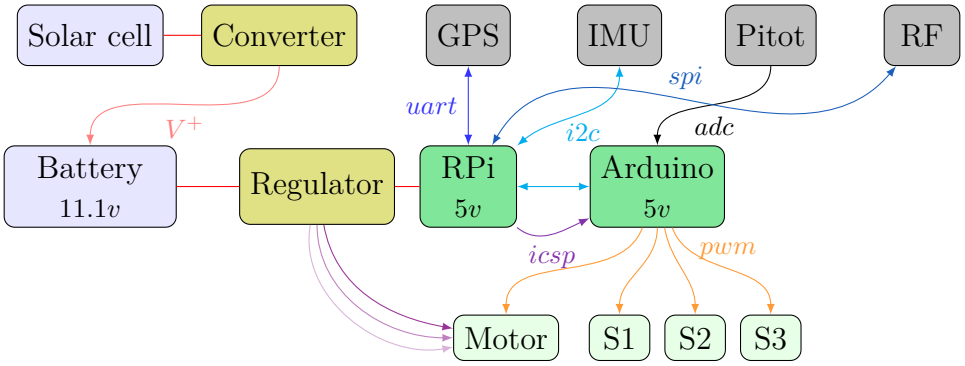
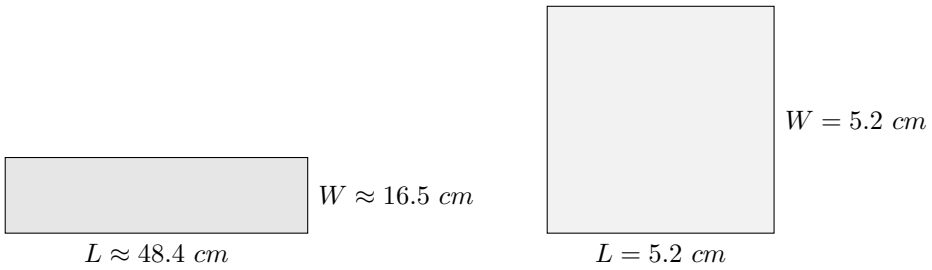


Figure 4.7: Connections of the complete system.

Solar Cells

Measures of glider wing are: length(L) $\approx$ 48.4cm and width(W) $\approx$ 16.5cm. In order to respect physical area, we are going to calculate the number of solar cells. A solar cell has a Max. Power of 0.43 W, 0.5 V, 0.45A and an efficiency of 17%.



(a) Wing (b) Solar cell

Figure 4.8: Dimensions of a wing and solar cell.

We calculate the number of solar cells: $\frac{Solar(L)}{Cell(L)} = \frac{48.4\ cm}{9.3077\ cm} = 9.3077$ cells, to be realistic we have 9 cells. Every cell produces $0.5\ v$, so we have $9{*}0.5\ v = 4.5\ v$. On the other hand, we calculate the number of lines, $\frac{Solar(W)}{Cell(W)} = \frac{16.5\ cm}{5.2\ cm} = 3.17307$ lines, again to be realistic we will have 3 lines. In theory, if we multiply the number of lines and total of voltage for each line, we will have $3 * 4.5\ v = 13.5\ v$ in one wing.

Our battery has $11.1\ v$, we must connect a protection system this will help for overcharging and discharging levels voltage. This module will charge our three batteries at the same time for having an equal voltage level. Nevertheless, all these modules need $12.6\ v$ as polarization. For that solar cells should provide a voltage more or equal that 12.6, supposing that we use previous solar cell dimensions as we calculated before. There are different configuration to connect solar cells, serial or parallel. To increase voltage serial connection is used. To

increase current parallel connection is used.

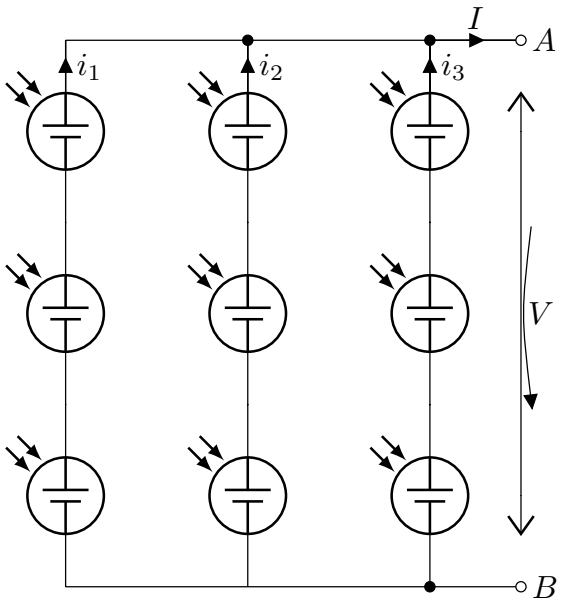
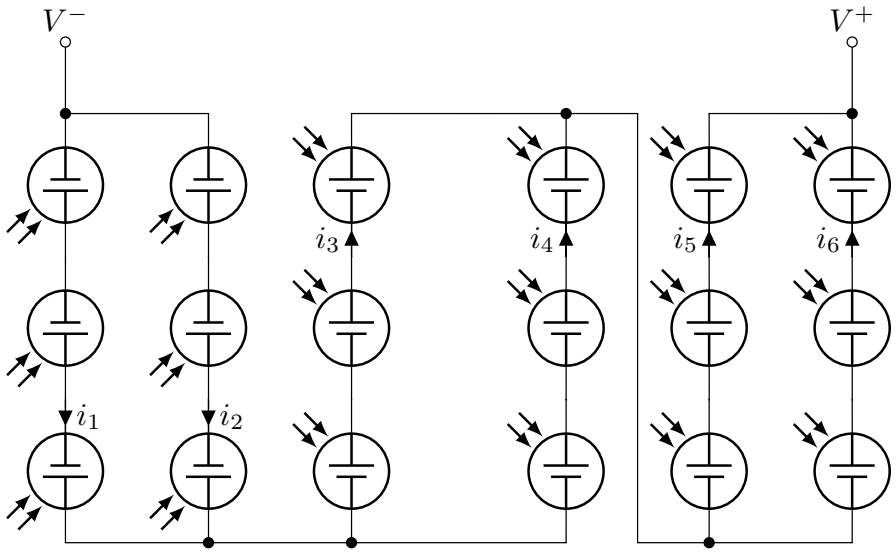
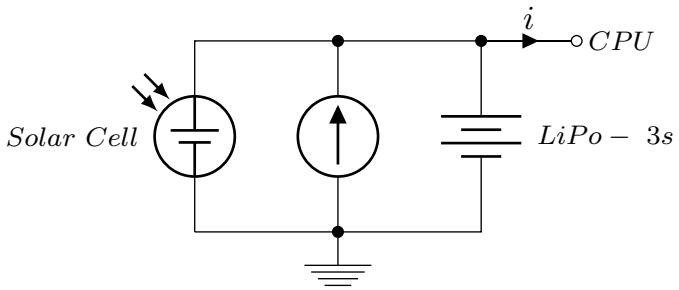


Figure 4.9: Connection of solar cells for a wing.

From Figure 4.8 is calculated a $4.5\ v$ and 3 lines, Figure 4.9. That is $V = 4.5\ v$ and $I = 3 * 0.45\ A = 1.35\ A$ for one wing. Of course, we can mix configuration to have different values of current and voltage. We should have $13.5\ v$ and $0.90\ A$, that will be good to charge batteries with system protection. That is connecting 9 cells in serial $(4.5v)$ and 2 lines in parallel $(0.90A)$, let's call this module A. Now, if we connect three modules, we will have $13.5\ v$ and $0.90\ A$, For simplification of calculations the reduction of the circuit can be replaced by a current source, Figure 4.10. Implemented autonomous power system this will allow us to aim to find thermal to recharge the batteries and continue flying. A glider is one of the most relevant aircraft, in terms of the ratio of the distance traveled and the loss of altitude. Making it more efficient to fly than other aircrafts. An autonomous motor-glider, a convenient choice(cost) in terms of weight and aerodynamics. The principle of a glider is that it uses the "vertical" wind (thermal and dynamic energy). It is to "climb", which that means to increase in altitude and reach an updraft, while "descent", expresses the rate of descent in a downward burst. "Zero" descent means that updrafts are strong enough to maintain flight, but not enough to allow climbing. As in the nature, there are various species of birds using the same principle. For example: albatross and condor. The scenario of the search for thermal springs is not monotonous, since to use a glider we must know in real time the climatological conditions which have a non-linear and unpredictable behavior, Figure 4.11. We think that using our resilient model unpredictable aspects could be captured, it would be interesting and motivating to be able to do something of this importance.



(a) Connection of solar cells in two wings.



(b) Simplification of the solar circuit.

Figure 4.10: Connection of solar cells in two wings and simplification of the circuit.

Articles

1. The first article deals with the description of the instruments that allow a pilot to take the pertinent actions to control an airplane. This information is represented in Reiter's logic allowing us to calculate fixed points. Eventually an operating system and data fusion are described (V. Medina, Pierre Siegel, and Andrei Doncescu 2017).

2. The second article talks about a take-off simulation where information as default and calculation of extensions are represented. These extensions are analyzed to take into account the effects of the combinations of actions (J.-L. V. Medina, Pierre Siegel, and Andrei Doncescu 2017).

3. The third article is focused on the use of the property of resilience to our non-monotonic model, having the ability of recovering in face of a contradiction. Theoretically, the desired behavior is shown in a discrete and

continuous representation (J. L. V. Medina, Pierre Siegel, and Andrei Doncescu 2018).

4. This article presents an intelligent and adaptable system. This definition covers any system where the knowledge base contains the objectives, situations and actions. It is an intelligent system because it proves assumptions with axioms. It is also an adaptable system because the property of resilience is considered. This property is included thanks to default logic, decision making model and Minsky's model. This last article was accepted for ICARCV 2018.

Autonomous Aerial Vehicle
Based on Non-Monotonic Logic

José Luis Vilchis Medina[1], Pierre Siegel[1] and Andrei Doncescu[2]
[1]*Aix Marseille Université, CNRS, LIF, Marseille, France*
[2]*LAAS, CNRS, Toulouse, France*

Keywords: Autonomous Motor-glider, Non-monotonic Reasoning, Default Logic, Decision-making, Solar Cells, Artificial Intelligence.

Abstract: In this article we study the case of an autonomous motor-glider. The aims of the aircraft is to maintain its flight as long as possible, taking advantage of the rising air from the ground, known as thermals, despite of limited energy resources and possible external influences, such as turbulences. The pilot task being to make decisions with incomplete, uncertain or even contradictory information, as well as driving to the desired path or destination. We propose the formulation of a model from the point of view of logical theory, using non-monotonic logic and more specifically default logic, to tackle these problems. Finally, we present the results of a simulation for further application in a glider(reduced model) which use solar cells for power management in embedded system.

1 INTRODUCTION

The glider is one of the most relevant aircraft, in terms of the ratio of the distance traveled and the loss of altitude. Making it more efficient to fly than other aircrafts. In this paper, we focus on an autonomous glider(reduced model), a convenient choice(cost) in terms of weight and aerodynamics. The principle of a glider is that it uses the "vertical" wind (thermal and dynamic energy). It is to "climb", which that means to increase in altitude and reach an updraft, while "descent", expresses the rate of descent in a downward burst. "Zero" descent means that updrafts are strong enough to maintain flight, but not enough to allow climbing. As in the nature, there are various species of birds using the same principle. For example: albatross and condor. To perform those flight maneuvers, a pilot must take decisions concerning the flight situation from the cockpit. He has access to different instruments showing altitude, wind speed, inclination of the aircraft, etc. Another constraint is that he also needs to find thermals, respecting the aeronautical and security regulations. Taking these considerations into account, the pilot must follow his desired flight path applying control commands. Increasing or decreasing altitude, turning to the right or left, etc. These rules are applicable to many types of aircraft. We introduce a discrete model of flight rules. This method is based on non-monotonic logic. There are different research proposals in non-monotonic logic reasoning: default reasoning, autoepistemic reasoning, reasoning in the presence of contradictory information and negative reasoning (El-Azhary et al., 2002). On another side, we can consider our model as a resilient system. These systems have the ability to resist and adapt from disturbances. They also are highly adaptive, having the capability to merge information, make decisions, interact with multiple agents and have a memory to facilitate learning (Goerger et al., 2014; Chandra, 2010). The main objective is to present a system based on flight rules capable to choose actions with incomplete information. The case study is presented in section 2. In section 3 classical logic representation is described. Section 4 is dedicated to the theoretical definition of default logic and its properties. Section 5 contains an explanation of the logical system using default logic representation and finally, section 6 practical case is described.

2 RELATED WORK

Research shows (Toulgoat et al., 2011; Toulgoat, 2011; Doncescu and Siegel, 2015; Le et al., 2013) that prove decision-making by non-monotonic logic is encouraging in the field of artificial intelligence. A

Medina, J., Siegel, P. and Doncescu, A.
Autonomous Aerial Vehicle - Based on Non-Monotonic Logic.
DOI: 10.5220/0006304002360241
In *Proceedings of the 3rd International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2017)*, pages 236-241
ISBN: 978-989-758-242-4

pilot is a person who has the function of guiding an aircraft in flight. Generally, the main cockpit flight control are: control yoke, rudder pedals and engine speed. Aircraft cockpit provides necessary information to control the trajectory and operation of the aircraft (de l'Aviation Civile., 1992).

2.1 Flight Indicators

On-board aircraft the basic set of instruments, also called "six pack"(de l'Aviation Civile., 1992). These six instruments are standard and many cockpits basic indicators are:

- *Altimeter*: shows the aircraft's altitude (in feet) above sea-level. As the aircraft ascends, the altimeter to indicate a higher altitude and vice versa.
- *Airspeed indicator*: shows the aircraft's speed (in knots) relative to the surrounding air. It works by measuring the ram-air pressure in the aircraft's Pitot tube relative to the ambient static pressure.
- *Vertical speed indicator*: sometimes called a variometer or also rate of climb indicator, senses changing air pressure, and displays that information to the pilot as a rate of climb or descent in feet per minute or meters per second.
- *Attitude indicator*: also known as an *artificial horizon*. Shows the aircraft's relation to the horizon. From this the pilot can tell whether the wings are level (*roll*, Fig. 1) and if the aircraft nose is pointing above or below the horizon (*pitch*, Fig. 1).
- *Turn indicator*: This instrument includes the Turn-and-Slip Indicator and the Turn Coordinator, which indicate rotation about the longitudinal axis.
- *Heading indicator*: displays the aircraft's heading with respect to magnetic north when set with a compass.

2.2 Flight Control

One time the pilot knows the flight situation through on-board instruments, he must take decisions and apply control commands(actions). Thereby accurately correcting the trajectory. For flight control, the pilot has 3 types of controllers. Generally, these controllers are in much type of aircrafts (de l'Aviation Civile., 1992). Description of controllers for our case, are described below:

- *Yoke*: has the function of controlling in both pitch and roll (Fig. 1).

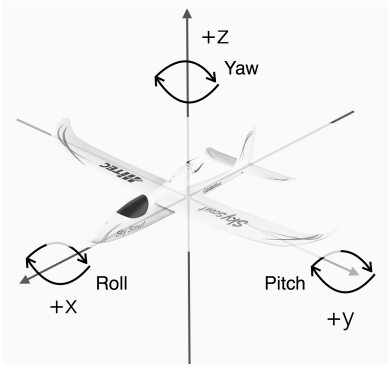


Figure 1: Definition of aircraft body axes.

  1. *Pitch controller*: When the yoke is pulled back the nose of aircraft rises and when it is pulled forward the nose descend.
  2. *Roll controller*: When the yoke is turned right the aircraft rolls to the right(turning right) and when it is turned left the aircraft rolls to the left(turning left).
- *Rudder*: also *pedal direction*. When pilot push left pedal, rudder deflects to the left and when pilot push right pedal, rudder deflects to the right. With rudder, pilot can not use it to turn left or turn right. Rudder is important for flight stabilization.
- *Engine power*: When pilot turn on the engine, it provides propulsion to the aircraft, if necessary.

Aircraft control systems are based on integro-differential equations or state-space form (Fossen, 2011). We present another method/solution tackled from the perspective of Artificial Intelligence. Next, we introduce how we can describe the issue by using logical representation.

3 CLASSICAL LOGIC REPRESENTATION

To describe actions, we use classical logic language L (Propositional or First-Order Logic). In L, we can represent, for example, $motor(t_i)$ to say that motor is active at the time t_i. Or $altitude(low, t_i)$ to say that the altitude is decreasing at the time t_i. We are in a logical framework, so it is possible to represent almost everything we want in a natural way. Classical logic, such as First-Order Logic is monotonous. What it means, $A \vdash w$ then $A \cup B \vdash w$. This is, by adding new information or set of formulas to a model, the set of consequences of this model is not reduced. The property of monotony is very important in the world of mathematics, because it allows to describe lemmas previously demonstrated. But this property cannot be

applied to uncertain and incomplete information. But in real world, non-monotonic is presents in many situations, this is, each new information we learn can modify or invalidate previous deductions. A classic example is:"Birds typically fly". This expression cannot be translated by the formula $bird(X) \rightarrow fly(X)$, which will signify that all birds fly. But, there are exceptions to this rule (i.e. if $X = Penguin$, by definition a penguin is a bird but he not fly). These exceptions are well known in Artificial Intelligence. In the next table, we represent the actions that pilot can do to correct the trajectory of the glider. We consider a discrete system with three possible actions for each of the controllers, except for the motor(which it has two states), resulting 11 states.

Table 1: Possible actions.

Roll:	*Pitch:*
$yoke(left,t_i)$ $yoke\_roll(neutral,t_i)$ $yoke(right,t_i)$	$yoke(pull,t_i)$ $yoke\_pitch(neutral,t_i)$ $yoke(push,t_i)$
Yaw:	*Propulsion:*
$rudder(left,t_i)$ $rudder(neutral,t_i)$ $rudder(right,t_i)$	$motor(t_i)$ $non\_motor(t_i)$

For the first approach, we will explicitly give basic pilot rules. For example, we might say, "the yoke is to the right position at the time t". We can represent in classical logic: $yoke(left,t_i)$. We use this notation: var = variometer, $ther$= thermal, alt = altitude, non_ther = non thermal, srh_th = searching thermal. A set of rules is explicitly given:

$$var(up,t_i) \wedge non\_motor(t_i) \rightarrow ther(t_{i+1}) \quad (1)$$

$$batt(low,t_i) \wedge alt(low,t_i) \rightarrow land(t_{i+1}) \quad (2)$$

$$non\_ther(t_i) \wedge alt(low,t_i) \rightarrow land(t_{i+1}) \quad (3)$$

$$non\_ther(t_i) \wedge alt(low,t_i) \rightarrow srh\_th(t_{i+1}) \quad (4)$$

$$non\_motor(t_i) \wedge var(up,t_i) \wedge turn(left,t_i) \rightarrow yoke(right,t_{i+1}) \quad (5)$$

$$non\_motor(t_i) \wedge var(up,t_i) \wedge turn(right,t_i) \rightarrow yoke(left,t_{i+1}) \quad (6)$$

$$var(up,t_i) \wedge non\_motor(t_i) \rightarrow alt(up,t_{i+1}) \quad (7)$$

$$batt(low,t_i) \wedge non\_motor(t_i) \rightarrow alt(down,t_{i+1}) \quad (8)$$

$$motor(t_i) \wedge yoke(pull,t_{i+1}) \rightarrow alt(up,t_{i+1}) \quad (9)$$

Intuitively, we can say,"the variometer is increasing at the time t", in logical representation is: $var(up,t_i)$. It is of course possible to add others rules(Aeronautics Legislation) to take into account relations between the real scenario and our logical system. These logical formulations before presented, are problematic because there is a conflict. If for example we have a set of formulas: $F = \{var(up,t_i), non\_motor(t_i), batt(low,t_i)\}$, and now, we infer (rule 7 and 8) F: $alt(up,t_{i+1})$ and $alt(down,t_{i+1})$, which is a contradiction. To solve this conflict, we use non-monotonic logic, more specifically, the default logic.

3.1 Definition of Non-monotonic Logic

It is a family of formal logic devised to capture and represent inference, reserving the right to retract deductions when new information is added. The first works in the field of non-monotonic logics began with the realization to precise characterization of defeasible reasoning. Among the pioneers of the field in the late 1970's were J. McCarthy (McCarthy, 1980; McCarthy, 1986), D. McDermott and J. Doyle, and R. Reiter(Reiter, 1980). Deriving in different non-monotonic logics(Sombé, 1990; Suchenek, 2006; Delgrande and Schaub, 2005; Delgrande and Schaub, 2003) reasoning: default reasoning(Reiter, 1980), autoepistemic reasoning, reasoning in the presence of contradictory information, counterfactual reasoning, priority reasoning, and negative reasoning(El-Azhary et al., 2002).

3.2 Definition of Default Logic

A default theory T consists of a set of facts W, which are formulas of First-Order logic and a set of defaults D, which are rules of inference (Reiter, 1980; Grigoris, 1999; Lukaszewicz, 1988; Lifschitz, 1999). The main representational tool is that of a default rule, or simply a default. A default is an inference rule of the form: $\frac{A(X) : B(X)}{C(X)}$, where $A(X), B(X), C(X)$ are well-formed formulas (First-Order Logic). Where $X = (x_1, x_2, x_3, ..., x_n)$ as a vector of free variables(non-quantified). $A(X)$ are the prerequisites, $B(X)$ are the justifications and $C(X)$ are the consequent. Intuitively, a default means:"if $A(X)$ is true, and there is no evidence that $B(X)$ might be false, then $C(X)$ can be true". Default rules are used to create extensions. These can be contemplated as a set of inferences. It is named normal default, if $B(X) = C(X)$.

3.3 Definition of Extension

When defaults are calculated, the number of formulas inferred in the knowledge base W increase. An extension of the default theory $\Delta = (D,W)$ is a set E

of logical formulas(Reiter, 1980). An extension must to verify following property: If d is a default of D, whose the prerequisite is in E, without the negation of its justification is not in E, then the consequent of d is in E. Formally, E is an extension of Δ if and only if:

- $E = \cup_{i=0}^{\infty} E_i$ with:
- $E_0 = W$ and for $i \geq 0$
  $E_{i+1} = Th(E_i) \cup \{C(X) | \frac{A(X) : B(X)}{C(X)} \in D, A(X) \in E_i \text{ and } \neg B(X) \notin E\}$

where $Th(E_i)$ is the set of formulas derived from E_i. The previous definition is difficult to apply in practice. Because $\neg B \notin E$ supposes E is known, but E is not yet calculated. In the case of normal defaults($B(X) = C(X)$), an extension is defined: E is an extension of Δ if and only if:

- $E = \cup_{i=0}^{\infty} E_i$ with:
- $E_0 = W$ and for $i \geq 0$
  $E_{i+1} = Th(E_i) \cup \{C(X) | \frac{A(X) : C(X)}{C(X)} \in D, A(X) \in E_i \text{ and } \neg C(X) \notin E_i\}$

where $Th(E_i)$ is the set of formulas derived from E_i. According to Reiter(Reiter, 1980), if all defaults are normal, it exists at least one extension. Extensions are defined by a fixed point.

4 DEFAULT LOGIC REPRESENTATION

Flight rules are described by a set of rules. For example, $alt(up)$ is a positive literal, means that the glider increases in altitude. The dynamic of the system can be described by $var(up,t)$, which means that the variometer at the time t is increasing. Clauses are the simplest type of formula. Formally, a clause is a disjunction of literals $l_1 \vee l_2 \vee l_3 \vee ... \vee l_n$. A Horn clause is a clause with a maximum of one positive literal. It's a formula defined as $(f_1 \wedge f_2 \wedge f_3 \wedge ... \wedge f_i) \rightarrow g$, where f_i and g are positive literal. Similarly, the formula defined as $\neg(f_1 \wedge f_2 \wedge f_3 \wedge ... \wedge f_i)$ is equivalent to the negative Horn clause(literals can not be true simultaneously). In default logic, we have two sets of rules. The set D of defaults and the set W of facts. For example, set W: $batt(good)$ that is an elementary fact says that battery has sufficient power. And set D: $\frac{batt(good) : motor(on)}{motor(on)}$, this default rule means, "Generally when battery has sufficient power, motor is on". Therefore, we introduce normal defaults representation.

4.1 Set of Default [D]

The set of inference rules D describes possibles actions, including incomplete and contradictory information. It represents a normal default. By using default logic, the rules(7, 8 and 9 in section 3) above could be expressed intuitively as:

7' If $var(up,t_i), non\_motor(t_i)$ are true, and if $alt(up,t_{i+1})$ is not contradictory, then $alt(up,t_{i+1})$ is true.

8' If $batt(low,t_i), non\_motor(t_i)$ are true, and if $alt(down,t_{i+1})$ is not contradictory, then $alt(down,t_{i+1})$ is true.

9' If $motor(t_i), yoke(pull,t_{i+1})$ are true, and if $alt(up,t_{i+1})$ is not contradictory, then $alt(up,t_{i+1})$ is true.

Taking the rules(7',8' and 9'), default logic representation is presented, $d = \frac{A(X) : C(X)}{C(X)}$.

$$d_1 = \frac{var(up,t_i) \wedge non\_motor(t_i) : alt(up,t_{i+1})}{alt(up,t_{i+1})}$$

$$d_2 = \frac{batt(low,t_i) \wedge non\_motor(t_i) : alt(down,t_{i+1})}{alt(down,t_{i+1})}$$

$$d_3 = \frac{motor(t_i) \wedge yoke(pull,t_{i+1}) : alt(up,t_{i+1})}{alt(up,t_{i+1})}$$

4.2 Set of Default [W]

The set of facts W represents accurate and non-revisable information. The facts are formulas always true. Unary clauses are elementary source of information. We can use such a clause to represent a mutual exclusions. For example, taking the rule "The motor-glider can not search a thermal and land at the same time", $\forall t, \neg(glider(search\_ther,t_i) \wedge glider(land,t_i))$.

4.3 Extensions Calculation

To illustrate the use of the default logic, we give in the next paragraph an example of calculation by using the Algorithm 1. $W = motor(off,t), alt(down,t), var(down,t), batt(good,t)$. With: $\forall t, \neg(glider(land,t+1) \wedge glider(srh\_th,t+1))$. Intuitively, W means that glider has not using motor(as propulsion), its altimeter is decreasing, its variometer is decreasing and its battery has sufficient power at time t_i, respectively.

The Algorithm 1 presented below is written in Prolog. Our algorithm calculates the extensions. As the clauses are Horn clauses, and as the defaults are normal. The results obtained using a set D(subsection 4.1) and a set W(subsection 4.2)

are: $E_0 = \{glider(srh\_th,t), motor(off,t), alt(down,t), var(down,t), batt(good,t)\}$

$$d_0 = \frac{alt(down,t) \wedge var(down,t) : glider(srh\_th,t+1)}{glider(srh\_th,t+1)} \quad (10)$$

$E_1 = \{glider(land,t), motor(off,t), alt(down,t), var(down,t), batt(good,t)\}$

$$d_1 = \frac{alt(down,t) \wedge var(down,t) : glider(land,t+1)}{glider(land,t+1)} \quad (11)$$

We can see that W uses a rule that says $glider(srh\_th,t+1)$ and $glider(land,t+1)$ are mutually exclusive. According to this rule, it is not possible to use d_0 and d_1 at the same time. If d_0 is used, $glider(land,t+1)$ is added to the extension, then mutual exclusive will cancel d_1. In the same way, if d_1 is used we cannot use d_0. We obtain 2 contradictory extensions(solutions). In the first one there is $glider(srh\_th,t+1)$(The glider can search a thermal at time $t+1$) and the other one there is $glider(land,t+1)$(The glider can land at time $t+1$). In this way, the conflict shown in section 3 is resolved.

Data: $E = \emptyset$ (Set of extensions E is empty)
Result: $E = \cup_{i=0}^{N} E_i$
Initialization;
CalculExtension(E);
while *there is a default* $(A(X) : C(X))/C(X)$ *that has not yet been inspected* **do**
  Select the default D;
  Verify $A(X)$ are true;
  Verify $C(X)$ is consistent with W;
  Add $C(X)$ to W;
end
Backtracking(deleting $C(X)$ added to W);
CalculExtension(E);
Algorithm 1: Calculation of extensions.

In practice, the choose of extension corresponds a weight to each default considering its importance. An example could be, to search a thermal is more priority (more weight) than land(Toulgoat, 2011; Grigoris, 1999). These decisions are based on weighted product model: $P(A_K/A_L) = \prod_{j=1}^{n}(a_{Kj}/a_{Lj})^{w_j}$ for $K,L = 1,2,3,...,m$. Using this method we ponder extensions then we select the best response.

5 APPLICATION

In this section we present the practical application and current work. In the Fig. 2 shows the different parts of our discrete model. On the left side in the

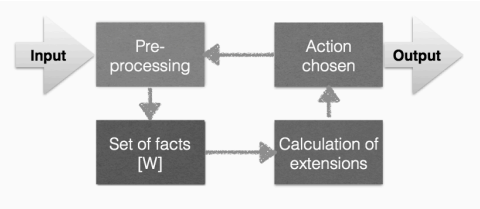


Figure 2: Operating diagram.

Fig. 2, we have the *Input* such as data, which accessible from electronic devices. Different data are airspeed, altitude, angles of pitch and roll, etc. Block *Set of facts[W]* accepts data for knowledge representation. Next, *Calculation of extensions* and *Action chosen* blocks are the representation of our Algorithm 1. Finally, we have *Output* such as an action to apply. Certainly to the servo-motors to control pitch, roll and eventually yaw(Fig. 1). These actions will be applied to a glider(reduced model,Sky Scout version R2GO). Our glider is equipped with a motor, in case he needs desperate increase his altitude or find a rising air. On the other hand, the motor-glider is equipped with an on-board computer. Prolog has been installed and sensors have been successfully implemented. The Algorithm 2 is presented which represents the blocks of *Pre-processing* and *Set of facts[W]*, described in Fig. 2. T_s is sampling time in seconds.

Data: From electronic device $[I]$
Result: Set of facts $[W]$
Initialization of electronics devices;
while T_s*=1* **do**
  CaptureData(I) from electronics devices;
  DescriptionFacts(W) by following our syntax;
end
Algorithm 2: Knowledge representation.

Data are from the Inertial Measurement Unit(IMU). An electronics device which collects angular velocity and linear acceleration data. Usually a combination of accelerometers and gyroscopes, sometimes also magnetometers. An IMU works by detecting changes in rotational attributes like pitch, roll and yaw, using gyroscopes. On the other hand, signal processing techniques(Fig. 2) has tremendous potential in the information fusion theory and in practical applications. Important part because is the process of integration of multiple data for knowledge representation. The acquisition of knowledge is defined by rules(subsection 4.3). In Fig. 3, rotation on the X axis is represented, the combination of accelerometer(circle) and gyroscope(line) data, the result is a signal(triangle) that we use to describe the world.

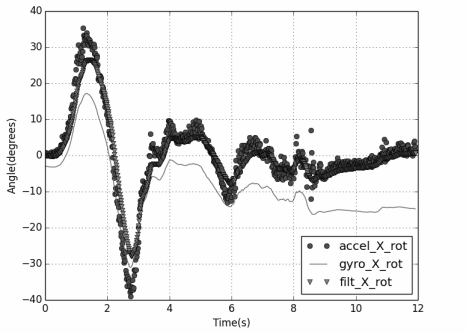


Figure 3: Data fusion.

6 CONCLUSIONS

In our approach, an algorithm based on default logic is proposed to tackle contradictory information. Flight rules are uncertain, because many situations include contradictions and we have to verify them. Furthermore, aeronautics legislation is based on contradictory rules. Non-monotonic logic has a great advantage in these kind of cases. We presented a simulation of a glider searching thermals. The results provide the basis for an autonomous motor-glider. We presented short and simple Prolog programs. For initial tests(around 100 rules), we calculate all extensions in a short time. The calculation takes 1956600 Logical Inferences Per Second (LIPS) and 0.049 seconds of central processing unit (CPU) time on a MacBook. Discrete model is currently being developed for testing. Sensors and on-board computer set-up is complete. We configured an embedded system, installing prolog (SWI-Prolog). We also have installed an embedded sensor with 10 degrees of freedom (DOF). To capture data, such as altitude, acceleration, pitch, roll, yaw, etc. for knowledge representation. More rules are currently incorporated to take into account the aeronautics legislation and solar power management for on-board systems.

ACKNOWLEDGEMENTS

We thanks to the Secretariat of Energy (SENER) through the Mexican National Council for Science and Technology (CONACYT)[grant number 581317/412566] for their support.

REFERENCES

Chandra, A. (2010). Synergy between biology and systems resilience. *Scholars' Mine*.

de l'Aviation Civile., D. G. (1992). *Manuel du pilote d'avion Vol à vue.*

Delgrande, J. and Schaub, T. (2003). On the relation between reiter's default logic and its (major) variants. *Seventh European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*.

Delgrande, J. and Schaub, T. (2005). Expressing default logic variants in default logic. *Journal of Logic and Computation*.

Doncescu, A. and Siegel, P. (2015). *DNA Double-Strand Break-Based Nonmonotonic Logic.* Computational Biology, Bioinformatics and Systems Biology. Elservier.

El-Azhary, Edrees, A., and Rafea, A. (2002). Diagnostic expert system using non-monotonic reasoning. *Expert Systems with Applications*, pages 137–144.

Fossen, T. (2011). Mathematical models for control of aircraft and satellites. *Department of Engineering Cybernetics, NTNU*.

Goerger, S., Madni, A., and Eslinger, O. (2014). Engineered resilient systems: A dod perspective. *Procedia Computer Science*, pages 865–872.

Grigoris, A. (1999). A tutorial on default logics. *ACM Computing Surveys*.

Le, T., Doncescu, A., and Siegel, P. (2013). Utilization of default logic for analyzing a metabolic system in discrete time. *ICCSA*.

Lifschitz, V. (1999). Success of default logic. *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*.

Lukaszewicz, W. (1988). Consideration on default logic: an alternative approach. *Comput. Intell.*

McCarthy, J. (1980). Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence*.

McCarthy, J. (1986). Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*.

Reiter, R. (1980). A logic for default reasonig. *Artificial Intelligence*, pages 81–132.

Sombé, L. (1990). *Reasoning under incomplete information in artificial intelligence: a comparison of formalisms using a single example.* John Wiley and Sons Inc.

Suchenek, M. (2006). On undecidability of non-monotonic logic. *Studia Informatica*.

Toulgoat, I. (2011). *Modélisation du comportement humain dans les simulations de combat naval.* PhD thesis, Laboratoire SNC/DCNS.

Toulgoat, I., Siegel, P., and Doncescu, A. (2011). Modelling of submarine navigation by nonmonotonic logic. *BWCCA*.

Contrôle de Vol d'un Planeur Basé sur une Logique Non-monotone

José-Luis Vilchis Medina[1] Pierre Siegel[1] Andrei Doncescu[2]

[1 2] Aix Marseille Université, CNRS, LIF, Marseille, France
[2] LAAS, CNRS, Toulouse, France

joseluis.vilchismedina@lif.univ-mrs.fr

Résumé

Le pilotage d'un avion consiste à faire divers choix d'actions pour réaliser une trajectoire. Mais les différentes phases de vol ont un environnement changeant, en conséquence on peut avoir des informations incertaines. Les situations peuvent changer en fonction des situations externes, telles que les perturbations atmosphériques, ou situations de sécurité et urgence du pilote. Le problème d'un pilote est alors contrôler l'aéronef dans des situations incertaines. Donc la commande de vol est un problème non-monotone. Nous présentons une méthode pour contrôler un planeur prenant en compte des facteurs d'incertitude et de contradiction.

Mots Clef

Logique non monotone, Logique des défauts, Représentation des connaissances, Planification, Moteur-planeur, Règles de pilotage.

Abstract

Aircraft piloting consist of making various choices of actions to realize a trajectory. But different phases of flight have a changing environment, therefore, We can have uncertain information. Situations may change depending on external conditions, such as atmospheric disturbances or safety and emergency situations of the pilot. The problem of a pilot is then to control the aircraft in uncertain situations. So flight control is a non-monotonic problem. We present a method for controlling a glider taking into account factors of uncertainty and contradiction.

Keywords

Non monotonic logic, Default logic, Knowledge representation, Planning, Motor-Glider, Rules of piloting.

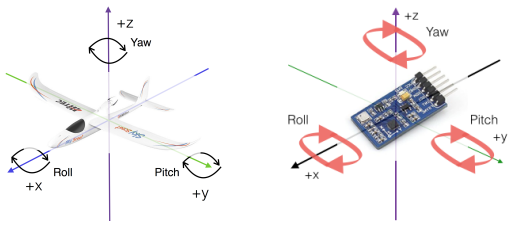
1 Introduction

La stabilisation des aéronefs, à différentes étapes du vol, est une partie importante pour une trajectoire bien définie. Il y a quatre forces qui maintiennent l'avion dans l'air : La force de portance, de traînée, le poids et la traction [2]. Une fois que l'aéronef est dans l'air, il doit s'orienter et se déplacer dans l'espace pour réaliser la trajectoire désirée. Généralment, les angles «roll», «pitch» et «yaw» sont utilisées pour mesurer l'orientation et l'inclinaison. Les avions ont besoin des moteurs comme agents propulseurs, cela permet d'avoir la vitesse nécessaire pour décoller et rester dans l'air [2]. Dans la plupart des cas, le contrôle de l'aéronef sont modélisés par des équations différentielles et des matrices de rotation [8]. Nous abordons le problème du point de vue de l'intelligence artificielle. Pour aborder cette problématique nous utilisons une théorie de la logique non-monotones. Nous utilisons un motoplaneur (planeur disposant d'un moteur d'appoint). L'avantage d'utiliser ce type d'aéronef, c'est parce que, il est l'un des meilleurs en termes de sa finesse (rapport entre la portance et la traînée). La finesse est liée aussi au rapport entre à la distance au sol et l'altitude perdue. Par exemple, s'il s'est déplacé de 40 km avec une perte d'altitude de 1 km, alors sa finesse est de 40. Dans les règles de pilotage, il y a des cas contradictoires ou incertains, si un pilote a une urgence (soit un problème technique, soit pour des raisons de sécurité, etc.), il doit violer certaines règles pour résoudre la problématique. Dans cet article nous allons présenter une méthode basée sur une logique non monotone pour la conduite d'un vol stable, prenant en compte l'incertitude et l'imprécision de l'information[1][7]. Nous présentons dans la section suivante la représentation des connaissances et la logique classique. La définition et les caractéristiques de la logique non monotone sont comprises dans la section 3. La représentation du système en logique des défauts est dans la section 4. Dans la section 5, nous décrivons une simulation d'une situation de vol avec toutes les solutions possibles et finalement les conclusions dans la section 6.

2 Représentation des Connaissances

La représentation des connaissances est une partie importante en intelligence artificielle car elle donne une description de l'environnement qui sera interpréter par un ordinateur. Dans ce contexte, grâce au capteur IMU [1] (Fig. 1), nous disposons de la mesure de différentes variables physiques (vitesses et accélérations, Fig. 2), avec lesquelles nous pouvons représenter les conditions spatio-temporelles du planeur. Ces informations vont permettre d'inférer les

1. Inertial Measurement Unit.



(a) Motoplaneur utilisé

(b) Capteur IMU avec 10 degrés de liberté

FIGURE 1 – Axes de reférence du planeur(a) et du capteur(b), les dessins n'ont pas à l'échelle.

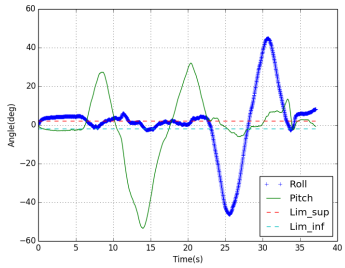


FIGURE 2 – Représentation dynamique du vol.

actions pour atteindre un objectif : décollage, atterrissage, tourner à droite ou à gauche, etc.

2.1 En logique classique

Avec la logique classique nous pouvons décrire des situations qui se passent dans la vie réelle ou encore représenter le monde. Dans la vie réelle, il y a des situations ou règles contradictoires. Si on prend l'exemple suivant, on peut représenter en logique classique le constat : "Lles ovipares pondent des œufs", avec la règle : $layEggs(X) \rightarrow ovipare(X)$. Maintenant, si nous considérons $X = Ornithorynque$, par définition, l'ornithorynque est un mammifère, or il pond des oeufs. En conséquence, la règle précédente ne peut pas être satisfaite. La logique classique est donc limitée, car elle ne prend en compte ni les aspects d'imprécisions ni ceux d'exceptions. Afin de traiter ce type de problématique, nous proposons d'utiliser la logique non monotone, plus particulièrement, la logique des défauts.

3 Logique Non-monotone

J. Mc-Carthy [5], D. Mc-Dermott et J. Doyle, et R. Reiter [6] ont été les premiers auteurs à travailler avec la logique non monotone dès la fin dès années 1970. Une logique monotone ne peut pas gérer les incertitudes (comme vu l'exemple présenté dans la section 2.1). La logique non-monotone est une logique formelle qui ne respecte pas la propriété de la monotonie, définie comme : $A \vdash w$, $A \bigcup B \vdash w$. Si on a un modèle A, et on ajoute des informations de B, on ne peut pas réduire les conclusions de w. Ainsi, elle permet de capturer et représenter le raisonnement par défauts (faits incertains et contradictoires). C'est un problème classique en Intelligence Artificielle. Une des logiques les plus connue et étudiée pour traiter cette problématique est la logique des défauts. C'est cette logique que nous présenterons dans la suite de ces travaux.

3.1 Logique des défauts

Une théorie des défauts T consiste en un ensemble de faits W, qui sont des formules en logique de premier ordre et un ensemble de défauts D, qui sont des règles d'inférence [6]. L'outil de représentation principal est la règle de défaut. Un défaut est une règle d'inférence sous la forme : $\frac{A(X) : B(X)}{C(X)}$, où $A(X), B(X), C(X)$ sont des formules bien formées (en logique de premier ordre). Où $X = (x_1, x_2, x_3, \ldots, x_n)$ est un vecteur de variables libres (non quantifiées). $A(X)$ est le prérequis, $B(X)$ est la justification et $C(X)$ est la conséquence. Il est possible d'utiliser la notation suivante, $A(X) : B(X) \rightarrow C(X)$. Intuitivement, une règle de défaut signifie : "Si $A(X)$ est vrai, et il n'y a aucune preuve que $B(X)$ soit faux, alors $C(X)$ est vrai ". Il est défini comme un défaut normal, si $B(X) = C(X)$. Les règles des défaut sont utilisées pour calculer des extensions.

3.2 Définition d'extension

Une extension de la théorie des défauts $\Delta = (D, W)$, est un ensemble E des formules logiques [6]. Une extension doit vérifier la propriété suivante : Si d est un défaut de D, dont le prérequis est dans E, sans que la négation de sa justification ne soit dans E, alors la conséquence de d est dans E. Formellement, E est une extension de Δ si et seulement si :

— $E = \cup_{i=0}^{\infty} E_i$ avec :
— $E_0 = W$ et pour $i \geq 0$

$$E_{i+1} = Th(E_i) \cup \{C(X) | \frac{A(X) : B(X)}{C(X)} \in D, A(X) \in E_i \text{ et } \neg B(X) \notin E\}$$

Où $Th(E_i)$ est l'ensemble de formules dérivés de E_i. La définition précédente est difficile à appliquer dans la pratique. Parce que $\neg B(X) \notin E$ suppose E est connu, mais E n'est pas encore calculé. Nous allons prendre la définition des défauts normaux, c'est-à-dire, $B(X) = C(X)$. Une extension est définie : E est une extension de Δ si et seulement si :

— $E = \cup_{i=0}^{\infty} E_i$ avec :
— $E_0 = W$ et pour $i \geq 0$

$$E_{i+1} = Th(E_i) \cup \{C(X) | \frac{A(X) : C(X)}{C(X)} \in D, A(X) \in E_i \text{ et } \neg C(X) \notin E_i\}$$

Où $Th(E_i)$ est l'ensemble de formules dérivés de E_i. Selon Reiter [6], si tous les défauts sont normaux, il existe au moins une extension. Les extensions sont définies comme points fixes ou solutions.

4 Représentation en Logique des Défauts

Dans cette section nous présentons les ensembles de faits W et de défauts D. Nous donnons également un exemple de calcul des extensions.

4.1 L'ensemble des faits W

W est un ensemble de faits vrais. Par exemple, $glider(decrease, t)$, signifie que le planeur a perdu de l'altitude au temps t par rapport au temps $t-1$. Dans cet ensemble, nous pouvons aussi décrire des exclusions mutuelles, ce sont des situations qui ne peuvent pas se produire en même temps. Par exemple pour les actions, le planeur ne peut pas tourner à gauche et à droite simultanément au même temps t. Formellement cette exclusion mutuelle est représentée de la façon suivante : $\forall t, \neg(yoke(pull, t+1) \wedge yoke(push, t+1))$.

4.2 L'ensemble des défauts D

L'ensemble des règles d'inférence D décrit des actions possibles en prenant en compte les informations incomplètes et contradictoires de l'environnement. Chaque règle est représenté sous la forme de défaut normal (Section 3.1). Par exemple, $\frac{glider(decrease,t) : yoke(pull,t+1)}{yoke(pull,t+1)}$. Intuitivement, cela signifie, "Si le $glider(decrease, t)$ est vrai, et si c'est possible que $yoke(pull, t+1)$, alors $yoke(pull, t+1)$". Autrement dit, si c'est possible de faire une action, on la ferra.

4.3 Exemple de calcul

Voici, nous avons une fonction F qui représente les faits vrais. Cette représentation d'écrit la situation du planeur au temps t (pour la fonction $glider(X, t)$, elle sera notée comme $g(X, t)$).

$$F : g(decrease, t), g(turn\_left, t), g(motor\_off, t), g(low\_altitude, t), g(low\_airspeed, t).$$

Nous pouvons dire en français que : le planeur perd de l'altitude, il tourne à gauche, il a le moteur coupé et une vitesse basse au moment t. Nous nous posons la question suivante : Quelles actions devons-nous faire pour avoir un vol stable ? Avec cette information, nous pouvons calculer les extensions (ensemble des actions) possibles pour atteindre l'objectif : vol stable. Ensuite, nous montrons les extensions avec les différents défauts obtenus (Fig. 3) :

$$E_0 = \{g(decrease, t), g(turn\_left, t), g(motor\_off, t), g(low\_altitude, t), g(low\_airspeed, t)\}$$

$$d_6 = \frac{g(decrease, t) : yoke(pull, t+1)}{yoke(pull, t+1)} \quad (1)$$

$$d_8 = \frac{g(decrease, t) : motor(on, t+1)}{motor(on, t+1)} \quad (2)$$

$$d_{14} = \frac{g(turn\_left, t) : yoke(right, t+1)}{yoke(right, t+1)} \quad (3)$$

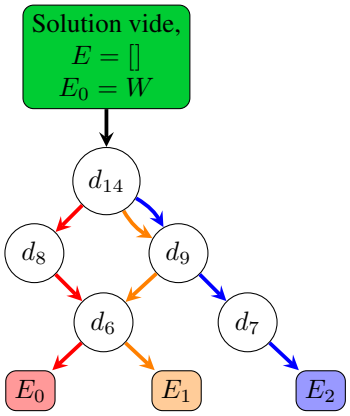


FIGURE 3 – Représentation graphique de calcul des extensions.

$$E_1 = \{g(decrease, t), g(turn\_left, t), g(motor\_off, t), g(low\_altitude, t), g(low\_airspeed, t)\}$$

$$d_6 = \frac{g(decrease, t) : yoke(pull, t+1)}{yoke(pull, t+1)} \quad (4)$$

$$d_9 = \frac{g(decrease, t) : motor(off, t+1)}{motor(off, t+1)} \quad (5)$$

$$d_{14} = \frac{g(turn\_left, t) : yoke(right, t+1)}{yoke(right, t+1)} \quad (6)$$

$$E_2 = \{g(decrease, t), g(turn\_left, t), g(motor\_off, t), g(low\_altitude, t), g(low\_airspeed, t)\}$$

$$d_7 = \frac{g(decrease, t) : yoke(push, t+1)}{yoke(push, t+1)} \quad (7)$$

$$d_9 = \frac{g(decrease, t) : motor(off, t+1)}{motor(off, t+1)} \quad (8)$$

$$d_{14} = \frac{g(turn\_left, t) : yoke(right, t+1)}{yoke(right, t+1)} \quad (9)$$

Nous pouvons constater que dans chaque extension calculée, nous avons des sous-systèmes cohérents [3][4]. Maintenant, nous avons résolu le problème de contradiction, présenté au début (Section 1). Nous pouvons ajouter des règles plus complexes pour une base de connaissances plus complète.

5 Simulation : Décollage

Dans cette section nous simulons une situation de décollage et nous calculons toutes les solutions possibles pour cette phase de pilotage. Les principaux éléments qui conduisent à l'objectif souhaité sont :

- État : C'est la représentation de la situation de l'objet (planeur) à chaque instant du temps.
- Temps : C'est l'instant entre les états.
- Action : C'est l'activité à réaliser afin d'atteindre l'objectif.

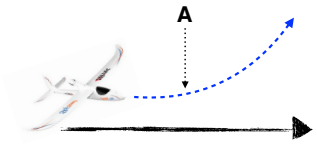


FIGURE 4 – Situation A.

Ces éléments sont clés pour inférer les actions suivantes. Dans le calcul sont considérés les trois aspects précédents. Dans cette situation (Fig. 4) nous avons les informations suivantes (pour la fonction $glider(X,t)$ vue précédemment, elle sera notée comme $g(X,t)$) :

$$G : g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t).$$

Si nous appliquons la théorie des défauts, plus particulièrement la logique des défauts (Section 3.1), nous obtenons 5 différentes extensions ou solutions.

$$E_0 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

$$d_{16} = \frac{g(roll\_stable,t) : yoke\_roll(neutral,t+1)}{yoke\_roll(neutral,t+1)} \quad (10)$$

$$d_{17} = \frac{g(pitch\_stable,t) : yoke\_pitch(neutral,t+1)}{yoke\_pitch(neutral,t+1)} \quad (11)$$

$$d_{20} = \frac{g(low\_altitude,t) : motor(on,t+1)}{motor(on,t+1)} \quad (12)$$

$$E_1 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

$$d_{16} = \frac{g(roll\_stable,t) : yoke\_roll(neutral,t+1)}{yoke\_roll(neutral,t+1)} \quad (13)$$

$$d_{17} = \frac{g(pitch\_stable,t) : yoke\_pitch(neutral,t+1)}{yoke\_pitch(neutral,t+1)} \quad (14)$$

$$d_{21} = \frac{g(low\_altitude,t) : motor(off,t+1)}{motor(off,t+1)} \quad (15)$$

$$E_2 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

$$d_{16} = \frac{g(roll\_stable,t) : yoke\_roll(neutral,t+1)}{yoke\_roll(neutral,t+1)} \quad (16)$$

$$d_{19} = \frac{g(low\_altitude,t) : yoke(push,t+1)}{yoke(push,t+1)} \quad (17)$$

$$d_{21} = \frac{g(low\_altitude,t) : motor(off,t+1)}{motor(off,t+1)} \quad (18)$$

$$E_3 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

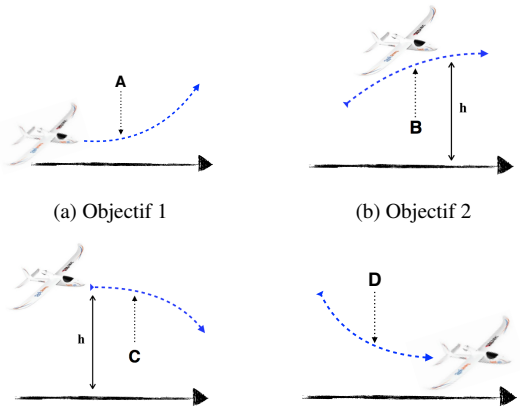
$$d_{17} = \frac{g(pitch\_stable,t) : yoke\_pitch(neutral,t+1)}{yoke\_pitch(neutral,t+1)} \quad (19)$$

$$d_{18} = \frac{g(low\_altitude,t) : yoke(pull,t+1)}{yoke(pull,t+1)} \quad (20)$$

$$d_{20} = \frac{g(low\_altitude,t) : motor(on,t+1)}{motor(on,t+1)} \quad (21)$$

$$E_4 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

$$d_{17} = \frac{g(pitch\_stable,t) : yoke\_pitch(neutral,t+1)}{yoke\_pitch(neutral,t+1)} \quad (22)$$



(a) Objectif 1 (b) Objectif 2

(c) Objectif 3 (d) Objectif 4

FIGURE 5 – Différents objectifs pendant le vol.

$$d_{18} = \frac{g(low\_altitude,t) : yoke(pull,t+1)}{yoke(pull,t+1)} \quad (23)$$

$$d_{21} = \frac{g(low\_altitude,t) : motor(off,t+1)}{motor(off,t+1)} \quad (24)$$

La meilleure extension et/ou solution de la situation A simulée précédemment (Fig. 4) est l'extension E_3 qui a les combinaisons des actions $\{yoke(pull,t+1), yoke\_pitch(neutral,t+1), motor(on,t+1)\}$ nécessaires pour atteindre l'objectif : **décollage**. Puisque l'extension E_0 a : $\{yoke\_roll(neutral,t+1), yoke\_pitch(neutral,t+1), motor(on,t+1)\}$, le résultat est un vol droit que ce n'est pas l'objectif. L'extension E_1 a : $\{yoke\_roll(neutral,t+1), yoke\_pitch(neutral,t+1), motor(off,t+1)\}$, il n'y a pas de mouvement sur le planeur car le moteur est coupé. L'extension E_2 a : $\{yoke\_roll(neutral,t+1), yoke(push,t+1), motor(off,t+1)\}$, même résultat que l'extension précédente, pas de mouvement car le moteur est coupé. Et finalement, l'extension E_4 a : $\{yoke(pull,t+1), yoke\_pitch(neutral,t+1), motor(off,t+1)\}$, même résultat que l'extension précédente, pas de mouvement car le moteur est coupé. Le choix d'une extension (ensemble des actions) est basé sur une analyse multicritère [3]. Nous avons choisi le modèle de produits pondérés car il permet de faire une analyse adimensionnelle sur les alternatives. Ensuite, nous montrons la définition :

$$P(A_K/A_L) = \prod_{j=1}^{n}(a_{Kj}/a_{Lj})^{w_j} \text{ pour } K, L = 1, 2, 3, ..., m.$$

Nous montrons sur la Fig. 5 les différents objectifs qui sont les différents étapes de pilotage.

6 Conclusion

La logique non-monotone est un outil important en Intelligence Artificielle car elle permet de capturer le raisonnement incertain. Nous avons fait une simulation d'un cas contradictoire (décollage) et nous avons constaté que la logique des défauts peut gérer les situations d'incertitudes et

des contradictions, de cette façon, nous avons obtenu 5 extensions ou solutions qui sont sous-systèmes cohérentes. Nous sommes en train d'améliorer les aspects des décisions, pour cela nous étudions l'optimum de Pareto. Le théorème de Pareto n'est pas sensible aux déséquilibres dans la répartition des ressources. Ce point est important pour résoudre le problème de gestion de l'énergie du système.

Références

[1] A. Doncescu, P. Siegel, DNA Double-Strand Break-Based Nonmonotonic Logic, *Computational Biology, Bioinformatics and Systems Biology*, 2015.

[2] Direction Général de l'Aviation Civile, *Manuel du pilote d'avion Vol à vue,* Direction Général de l'Aviation Civile, 1992.

[3] I. Toulgoat, Modélisation du comportement humain dans les simulations de combat naval, *Thèse de doctorat*, 2011.

[4] I. Toulgoat, P. Siegel, A. Doncescu, Modelling of Submarine Navigation by Nonmonotonic Logic, *BWCCA*, 2011.

[5] J. McCarthy, Circumscription - A form of non-monotonic reasoning, *Artificial Intelligence*, 1980.

[6] R. Reiter, A logic for Default Reasonig, *Artificial Intelligence*, pp. 81-132, 1980.

[7] T. Le, A. Doncescu, P. Siegel, Utilization of Default Logic for Analyzing a Metabolic System in Discrete Time, *ICCSA*, 2013.

[8] K. Terrell, S. Zein-Sabatto, Intelligent reconfigurable control system for aircraft flight control, *Southeast-Con, 2017, IEEE*, 2017.

Non-monotonie et Resilience: Application au Pilotage d'un Moto-planeur Autonome

José Luis Vilchis Medina[1] **Pierre Siegel**[1] **Andrei Doncescu**[2]

[1]Aix-Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
[2] LAAS, CNRS, France
{joseluis.vilchismedina,pierre.siegel}@lis-lab.fr, andrei.doncescu@laas.fr

Résumé

Dans cet article, nous présentons un système résilient pour un moto-planeur, basé sur une logique non-monotone. Les systèmes écologiques, biologiques et autres sont résilients, par exemple, les catastrophes naturelles, les bancs de poissons et les nuées d'oiseaux. C'est une propriété qui permets d'absorber les perturbations et de surmonter les adversités. Le pilotage est également un système résilient car il peut avoir des situations conflictuelles et l'environnement est imprévisible. Le pilote doit alors changer de comportement. Quand un pilote fait face à ce genre de situations, il entre dans un raisonnement incertain, malgré le fait qu'il doive prendre des décisions pour différents objectifs. Nous introduisons un modèle résilient non-monotone pour piloter un moto-planeur autonome. Ce modèle n'inclut pas la notion de temps. La logique des défauts a été utilisée pour trouver des points fixes à partir d'informations ambiguës et conflictuelles. Le modèle proposé ici contient une connaissance du monde avec un ensemble de situations, d'objectifs et d'actions. Après le calcul des solutions plausibles, la prise de décision est basée sur une théorie non-probabiliste. Nous avons défini une notion de stabilité dans des situations de pilotage incertaines en utilisant la propriété de résilience.

Abstract

This article presents a resilient system for a motor-glider based on non-monotonic logic. Resilience is the property of a system allowing to absorb disturbances and overcoming adversities. Ecological, biological and many other systems are resilient, for instance, natural disasters, fish school and birds flock. Piloting is also a resilient system because it could have conflicting situations and environment is unpredictable, so behavior change. When a pilot faces such kind of situations, he enters into uncertain reasoning, despite the fact that he must take decisions for different objectives. We introduce a non-monotonic resilient model to pilot an autonomous motor-glider. This model does not include the notion of time to make decisions. Default logic is used to find fixed points from ambiguous and conflicting information. The resilient model proposed here contains a world knowledge with a set of situations, objectives and actions. After computation of plausible solutions, decision-making is based on a non-probabilistic theory. We define a notion of stability in uncertain situations of flight using the property of resilience.

1 Introduction

After many decades of research in the field of aeronautics today new directions open to use plane or any aerial vehicle for different applications in an autonomous way. Definition of autonomous aerial vehicle (UAV) is a programmed vehicle which receives directions from a source placed at a distance apart from it. It can be flown without a pilot by using a particular system on the ground. Using UAV is not only confined to the military fields. They are used in many areas such as agriculture, construction, entertainment, and so far. In our study, we are using a motor-glider. Motor-glider has many constraints when is flying, besides, the pilot has short time to make decisions. He considers certain information and thus be able to make actions, for instance, increase or decrease the engine power, turn the steering wheel to the right, pull, etc. A motor-glider is equipped with an engine motor, which allows to take-off and climb without assistance, in contrast with a normal glider that is non-motorized. In the 70's, Holling introduced the term of resilience to model the dynamics of natural disasters [5]. In other fields of science the concept of resilience is defined as the property of a system to absorb and anticipate perturbations [4]. In ecology, resilience aids to understand natural disasters behavior [5, 1]. In engineering, resilience ensures consistency, robustness and stability [5, 14], even in uncertain environments [16]. Piloting use non-monotonic reasoning when environment change, it should take decisions because perturbations appear. In

this paper, we present a model based on non-monotonic logic and the property of resilience, both we will allow us to tackle the problem of uncertain reasoning with incomplete information and stability of an autonomous motor-glider. The sections are composed in the next order. First, the traffic pattern circuit for airplanes and states of flight are explained. Non-monotonic reasoning and default logic are presented in section 3. The complete model as well as situations, objectives and actions are described in section 4. The properties of resilience and stability are explained in section 5. The implementation of the model is described in section 6. Finally, conclusion is described in section 7.

2 Traffic Pattern Circuit

Every pilot knows the traffic pattern circuit. It is one of the basic maneuvers to take-off and land. However, it contains the necessary rules to carry out a long flight. Next, we explain the different states of flight for an airplane. Since most of these states are the same for motor-gliders. First of all, the pilot needs to know airplane states, so he uses the cockpit. The cockpit is a set of instruments on board that displays parameters such as ***airspeed*** (Miles/h), ***artificial horizon*** (pitch and roll) [1], ***variometer*** (Feet/s), ***altitude*** (Feet), ***compass***,...

Traffic pattern circuit, Fig. 1, has different flight stages. It starts at the point S_p where the airplane is in *Rest*. When the pilot is ready and he has the authorization, he increases all the engine power to get a right airspeed to take-off (point a). This is airplane should climb to a suitable height (point b). After that, the pilot should turn the yoke to the left making an orthogonal path to the runway (point c). At this point, he turns again the yoke to the right having constant airspeed, constant altitude and zero vertical speed. When he arrives to the point d he will prepare to land. Turning the yoke to the right, decreasing in altitude and having negative vertical speed, until arrive to the point e. Once again, he should turn the yoke to the right to continue decreasing in altitude and having a stable roll and negative pitch, until point f. After this point, airplane touches the ground. Final point F_p is where airplane state is again in *Rest*. In order to formalize the circuit, we represent knowledge using First-Order Logic (FOL). This is a formal language, which allows to represent almost everything in natural sense, it is expressiveness. We could say : "An airplane is landing". Using FOL, we have : $land(airplane)$. Another instance could be : "Pilot increase the engine power". In FOL we have : $engine(pilot, increase)$ and so on. The flight manual contains all necessary information to pilot an airplane, including technical descriptions, physical limitations, rules and emergency procedures. But all these information are generals that depending on the situation could have contradictory rules. For instance, there is a rule that says the minimum over flight height will never be less than 500 feet, in fact this altitude depends of the agglomeration. This rule could be expressed in FOL, considering that x = airplane, as follows :

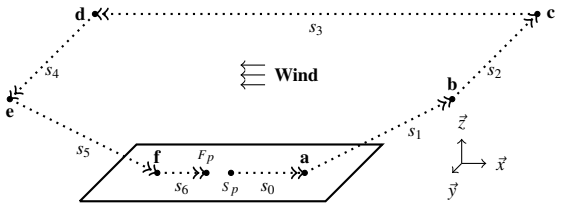


FIGURE 1 – Traffic Pattern.

$$altitude(x) \rightarrow (x > 500) \quad (1)$$

But when an airplane lands its altitude is less than 500 feet. This could be expressed as follows :

$$land(x) \rightarrow (x < 500) \quad (2)$$

General rules are described in the flight manual but pilot is the one who finally decides if he violates such rules. We can see that equation (1) and (2) are contradictory. This is a limitation of classical logic, because it is monotonous. Formally the property of monotony is : $A \vdash w$ then $A \cup B \vdash w$. In other words, adding new information to a model, the consequences are not reduced. This kind of problem, about contradictions and exceptions is well known in Artificial Intelligence. It has been studied from along time [10, 9]. We can see that it is a non-monotonic problem. In order to tackle it we have to move from this framework of classical logic. Because pilot use non-monotonic reasoning when he has new information and he can break the rules.

3 Non-monotonic Reasoning and Default Logic

Non-monotonic reasoning is a class of reasoning where we make assumptions about things jumping to the conclusions. Humans use kind of reasoning, this is the way we can do in situations with incomplete and contradictory information. Pilot does the same thing because environment change and he will have exceptions. In the 1970s, J. McCarthy, D. McDermott, Reiter and others started studies on non-monotonic inference, deriving in default reasoning, autoepistemic reasoning and more others. A robust formalization with exceptions is that Reiter proposed, default logic [12]. In default logic, a default theory is a pair $\Delta = (D, W)$, where D is a set of defaults and W is a set of formulas strictly in FOL. A default d is : $\frac{A(X):B(X)}{C(X)}$, where $A(X), B(X), C(X)$ are well-formed formulas. Where

1. ***Pitch*** is the angle formed by the airplane when has rotated around "y-axis". Similarly, ***roll*** is formed by the airplane when has rotated, but around "x-axis".

$X = (x_1, x_2, x_3, \ldots, x_n)$ is a vector of free variables(non-quantified). $A(X)$ are the ***prerequisites***, $B(X)$ are the ***justifications*** and $C(X)$ are the ***consequences***. Intuitively a default means,"if $A(X)$ is true, and there is no evidence that $B(X)$ might be false, then $C(X)$ can be true". When defaults are used it means extensions are calculated. An extension of a default theory Δ is a set E of logical formulas [12] with the smallest set that must verify the following property : If d is a default of D, whose the prerequisite is in E, without the negation of its justification is not in E, then the consequent of d is in E.

Definition 1. E is an extension of Δ iif :
— $E = \bigcup_{i=0}^{\infty} E_i$ with :
— $E_0 = W$ and
— for $i > 0$, $E_{i+1} = Th(E_i) \cup \{C(X) \mid \frac{A(X):B(X)}{C(X)} \in D, A(X) \in E_i \wedge \neg B(X) \notin E\}$
where $Th(E_i)$ is the set of formulas derived from E_i.

The previous definition is difficult to compute in practice. Because $\neg B \notin E$ supposes that E is known, but E is not yet calculated. In the case of normal defaults, $B(X) = C(X)$, E is an extension of Δ iif : we replace $\neg B(X) \notin E$ by $\neg C(X) \notin E_i$. According to Reiter if all defaults are normal, it exists at least one extension. Extensions are defined such as fixed points.

Example 1. Using default logic, from predicates (1) and (2) we have 3 defaults, which contains general information about altitude, where $alt = altitude, x = airplane, std\_fgt = steady\_flight$:

$$d_1 = \frac{((alt(x) > 500) \wedge roll(x, stable)) : std\_fgt(x)}{std\_fgt(x)} \quad (3)$$

$$d_2 = \frac{((alt(x) \leq 500) \wedge roll(x, stable)) : land(x)}{land(x)} \quad (4)$$

$$d_3 = \frac{(land(x) \wedge obstacle) : climb(x)}{climb(x)} \quad (5)$$

In natural sense, d_1 describes if x has an altitude more than 500 feet with a stable roll, and it is possible that x is in a steady flight, then x is in a steady flight. Default d_2 describes that if x has an altitude less than 500 feet with a stable roll, and it is possible that x lands, then x lands. And default d_3 describes if x lands and there is an obstacle, and it is possible to climb, then x climbs. Now, we are going to use these three defaults assuming that we have the following information :

$$W = \{(alt(x) \leq 500), roll(x, stable), obstacle\} \quad (6)$$

From $\Delta = (D, W)$, we calculate the set of extensions. We find $E_1 = W \cup land(x)$, where x lands, by using the default d_2. On the other hand, we find $E_2 = W \cup climb(x)$, where x climbs, by using the default d_3. We have two coherent solutions. Solving the problem of contradictory information.

There are mandatory rules that cover flight physics, security and more. For instance, in case of engine failure, x lands. Or if there is an obstacle in the runway, x must not land. But if x has a fault, the pilot must land to not die, so the risk is huge. In this case, the extension calculated will have a high weight. When different solutions are computed we should take into account criteria such as emergency, security, regulation, energy, etc. to choose the better decision. Situations are constantly changing because environment change. Using probabilities to choose one of them it is not the idea, such as Weighted Product Model or Weighted Sum Model [15]. We propose another manner to make decisions from a different point of view. We are in uncertain framework, we consider a non-probabilistic model. We focus on the opportunist model [2]. This model creates an opportunistic loss (or regret) matrix [13]. Formally, the set of regrets is defined as :

$$\forall E, \exists\, m_r = min\,\{max\,(c_i) - c_j\} \quad (7)$$

Where m_r is the minimization of the difference between the maximum value of the criteria c_i and alternatives c_j, this is for all extensions.

Example 2. Let us consider extensions and criteria. Criteria are information about the system or environment. Having two extensions, E_0 and E_1. E_0 has a higher value of

Table 1 – Criteria Table.

EXTENSION	**CRITERIA**	
	Energy	*Risk*
E_0	5	2
E_1	2	3

energy than E_1, that is, E_0 has a good status of battery or gas, for example. On the other hand, E_0 is less dangerous that E_1, in terms of risk (e.g. agglomeration). Regrets are calculated, $E_0 : \{0, 1\}$ and $E_1 : \{3, 0\}$. In order to obtain the better decision of these two extensions. E_0 is the better option which minimize the risk (m_r), which make sense, in real-life if an airplane has enough energy and pilot makes actions that are not dangerous, he will choose them.

Until now we present how to solve a problem with incomplete and contradictory information as well as the way to choose the better option when we have several extensions. In the next section we are going to introduce new concepts, taking into account the property of resilience and non-monotonic to reason.

4 Non-monotonic Model

When a pilot has a disturbance of any kind, he will naturally move away from the **objective** (O), this it could be

land, take-off, climb...However, he must make actions to achieve the goal. Pilot is in constant revision of behavior, taking information from the cockpit, from the environment and even from the control tower. Additionally, pilot should respect air regulations and navigation laws. For a better understanding, we introduce the following concepts.

4.1 Situations, Objectives and Actions

Firstly, the set of *situations* (S) contains information about parameters of the airplane (altimeter, airspeed, variometer...), environment, etc. On the other side, the set of *actions* (A) are what the pilot does physically (increase or decrease the engine power, turn the yoke to the left or right, ...) to the airplane. In this context, the ***situations*** and ***actions*** are represented by positive literals. We consider that for a certain situation, the challenge is to calculate the extensions that contain actions which allow to approach the desired ***objective*** (O). For instance, when an airplane is placed at the start point (S_p), Fig. 1, assuming it has the authorization, and it is possible to take-off, then the plane take-off. This objective could be described by a default as follows :

$$\frac{(rest(x) \wedge authorization) : takeoff(x)}{takeoff(x)} \tag{8}$$

In the same way, we could describe when a plane (starts at some point a) wants to maintain an altitude greater than 1500 feet with a north direction, to reach to the point b. A default could be as follows :

$$\frac{((alt(x) > 1500) \wedge compass(x, north)) : point(x, b)}{point(x, b)} \tag{9}$$

These are just two defaults as examples, but we can include many others in O. We consider two kind of objectives, short and long-term. The ***short-terms*** occur when there are perturbations and airplane moves away from the long-term objective. Thus, pilot will find another short-goal to get closer and converge. For instance, when the airplane is climbing (from the point a to the point b, Fig. 1) to an altitude of 1500 feet and there are wind disturbances, equation (8) is considered a sub-goal. On the other hand, a ***long-term*** objective is, for instance, maintain a steady flight for 5 minutes with an altitude of 1500 feet, equation (9) is considered a long objective.

Definition 2. In the world **K**, there is always a resilience trajectory R.

$$\forall S, \forall O, \forall A \subseteq K \; \exists R \tag{10}$$

Short-term objectives have very fast change in comparison with long-term. Nevertheless, short-terms will allow to achieve long-term. As the system evolves and disturbances appear, exploration is an important stage of the model. Because this part it is the main process to find different sub-goals that will allow to absorb the shock ζ. Sub-goals g are related to the extensions since they contain actions to converge to the final goal. It is so that the system can jump between sub-goals and have a resilient behavior. If

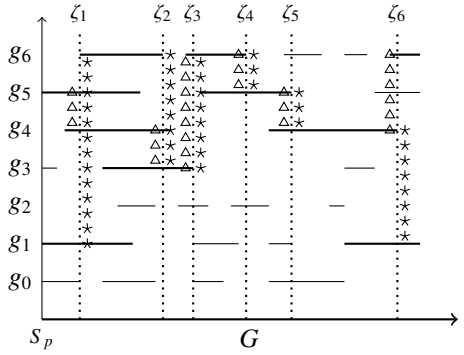


FIGURE 2 – Evolution of a goal G, switching sub-goals g when a disturb ζ occurs : $\triangle$ and $\star$ are the trajectories created.

we take a look at the discrete representation, Fig. 2. We have at the begging S_p the computation of four extensions : $\{g_0, g_1, g_3, g_5\}$, according to our decision-making model g_1 is chosen and then the system interacts with the environment. At some point, disturb ζ_1 occurs and extensions are computed one more time : $\{g_1, g_4, g_5, g_6\}$, the better solution is g_6 and then interaction happens again. This process occurs every time disturbs ζ appear. In this sense, computing and choosing extensions, trajectories ($\triangle$, $\star$) are created. For the first resilient trajectory $\triangle$, we have : $R_\triangle = \{g_5, \zeta_1, g_4, \zeta_2, g_3, \zeta_3, g_6, \zeta_4, g_5, \zeta_5, g_4, \zeta_6, g_6, \ldots\}$ and for the second trajectory ($\star$), we have : $R_\star = \{g_1, \zeta_1, g_6, \zeta_2, g_3, \zeta_3, g_6, \zeta_4, g_5, \zeta_5, g_4, \zeta_6, g_1, \ldots\}$

4.2 Model

We present a model that describes the evolution of pilot reasoning. In reality pilot makes two movements, he observes the horizon and next the cockpit, after that he does actions, he repeats this over and over again. This dynamic could be represented such as Fig. 3. The model has transitions but the notion of time is not considered. For example, if we have a situation s_i and if it is possible to go to the situation s_{i+1}, we should do actions. Firstly, we start with

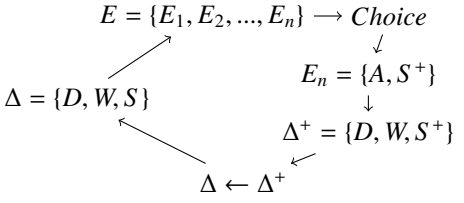


FIGURE 3 – Reasoning of a pilot based on default logic.

a default theory $\Delta = (D, W, S)$. Where D are the set of defaults, W are the set of FOL and S are the parameters of the airplane, environment, control tower, etc. We are considering that Δ is a default theory before a transition and Δ^+ is a default theory after a transition. Similarly, S is a situation observed before a transition and S^+ is a situation observed after a transition. From Δ, the set of extensions E is computed. Each extension contains actions. Once we have the solutions we must choice the better extension that brings us closer to the goal, then decision-making is based as before. After pilot applies actions he takes observations again (cockpit and environment) passing information from S to S^+. Then it goes back to Δ to compute extensions and choose the better one again. Sometimes for an airplane it is impossible to converge to the desired goal and alternative objectives must be found. In the set of objectives (O) the property of resilience is carried out. This is the property will be described in the next section.

5 Resilience

The concept of resilience is defined as the property of a system to absorb and anticipate perturbations [4]. So to apply this property to our model, let consider a knowledge world K, Fig. 4, which contains the set of *situations* (S), *objectives* (O) and *actions* (A). Inside the world K we will study the property of resilience. Since we know that the logic model has an evolution, we are interested to study its form and properties. We can define a trajectory as the satisfaction of 4 main properties : ***reorganization*** (α), ***exploration*** (β), ***release*** (γ) and ***conservation*** (δ), Fig. 4 [6, 14, 16]. We consider that *Non-monotonic reasoning* is exploration, *Choice* is reorganization and conservation, and finally interaction with environment is release. Trajectory has a form as closed loop that converge a stable equilibrium [5]. In control theory [8] stability is defined as follows :

Definition 3. A non-linear time-invariant system with $x' = f(x)$, $f : \mathbb{R}^n \to \mathbb{R}^n$. It has a point $x_e \in \mathbb{R}^n$ is an equilibrium point of the system if $f(x_e) = 0$. It is global asymptotically stable, if for every trajectory $x(t)$, we have $x(t) \to x_e$ as $t \to \infty$. It is locally asymptotically stable near or at x_e if there is $R > 0$, s.t. $\|x(0) - x_e\| \leq R \Rightarrow x(t) = x_e$ as $t \to \infty$.

We consider Lyapunov's definition for our study. For every short and long-term objective **O** a neighborhood (ϵ) of the exact point of convergence is defined as follows :

$$0 < \|\mathbf{O}\| < \epsilon \tag{11}$$

In our model, stability will be when every objective **O** is inside ϵ. In this context, we define : $R \equiv O$, where R is the theoretical trajectory of resilience and O is the trajectory of objectives of our model [6]. If the equivalence is valid, then the system will have stability in term of resilience. Theoretical trajectory is defined as follow : $R : \{\ldots \alpha, \beta, \gamma, \delta, \alpha, \ldots\}$. An interesting point of the model is if we increase the number of defaults, we will increase the degrees of freedom. This is an important remark, because we consider degree of freedom a space in **O** where it could pass a trajectory.

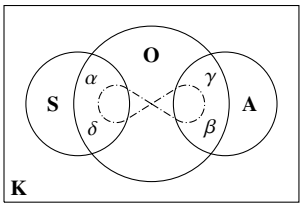


FIGURE 4 – Non-monotonic Resilience Stability.

6 Implementation

We are using a reduced-size model of motor-glider with a wingspan of 1366mm (53.75 in.), an overall length of 977mm (38.5 in.) and a HBM 2812-1100 Brushless Motor. On board a microcomputer based on Linux operating system is installed, it has the next characteristics : a cpu running at 1 GHz ARM11 (single core), 512 Mb of RAM and power consummation of 0.8 Watts. The microcomputer contains physical digital ports with serial communication protocols which allow to connect different devices. The inertial sensor provides the accelerations, angular velocities and measurements of the earth's magnetic field. These three information, it allows us to know the orientation of the motor-glider in space, for instance, if it is going up, down, turning...Altitude is provided by the GPS module but also it is calculated by an atmospheric pressure sensor. Pitot tube is an instrument that allows to measure the static and dynamic pressure, and thus to know the airspeed of the airplane, based on the Bernoulli's equation. For obstacle detection an ultrasonic sensor is used, with a max. detecting distance of 4-5m. The aileron control is done by servomotors through PWM signal. The circuit on board is supplied with 11.1 Volts and 1300 mAh LiPo battery. In the microcomputer, SWI-Prolog was installed. Until now, we have 80 defaults and the extensions are calculated in the order of milliseconds. However, if we increase the number of defaults, the calculation time does not increase much, since horn clauses are used [7, 3].

7 Conclusion

We introduced a resilient model for an autonomous airplane, using non-monotonic logic, in particular, default logic. We tackled contradictory and incomplete information to manage aviation rules and make decisions. We defined

a non-probabilistic model to choose an extension considering criteria such as security, energy, emergency...We used the property of resilience to find alternative solutions, when disturbs occur, and converge to the objective or sub-objectives. We described stability using Lyapunov's definition. The implementation is currently in progress with good results. This is a motivation to have a resilient model able to find thermal and to be able to fly as long as possible autonomously. We are also interesting to study Minsky's model, Fig.5, which describes how the mind get goals by changing the set of axioms in use [10, 11]. From our model in the Fig.3, we could consider the "Now" such as the actual situation S and "Want" such as S^+, the long-term objectives. The differences will be the actions that we should do to converge to the main objective.

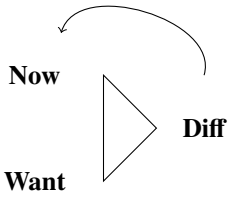


FIGURE 5 – Minsky's model.

Acknowledgements

We thanks to the Secretariat of Energy (SENER) through the Mexican National Council for Science and Technology (CONACYT)[grant number 581317/412566] for their support.

Références

[1] Chandra, Ashik: *Synergy between biology and systems resilience*. Mémoire de maîtrise, MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY, 2010.

[2] Chiles, Todd H et John F McMackin: *Integrating variable risk preferences, trust, and transaction cost economics*. Academy of management review, 21(1) :73–99, 1996.

[3] Doncescu, A. et P. Siegel: *DNA Double-Strand Break-Based Nonmonotonic Logic*. Computational Biology, Bioinformatics and Systems Biology. Elservier, 2015.

[4] Goerger, Simon R., Azad M. Madni et Owen J. Eslinger: *Engineered Resilient Systems : A DoD Perspective*. Dans *Procedia Computer Science*, tome 28, pages 865–872. ELSEVIER, 2014.

[5] Holling, Crawford S: *Resilience and stability of ecological systems*. Annual review of ecology and systematics, 4(1) :1–23, 1973.

[6] Holling, Crawford S: *Understanding the complexity of economic, ecological, and social systems*. Ecosystems, 4(5) :390–405, 2001.

[7] Le, T., A. Doncescu et P. Siegel: *Utilization of Default Logic for Analyzing a Metabolic System in Discrete Time.* ICCSA, 2013.

[8] Lyapunov, Aleksandr Mikhailovich: *The general problem of the stability of motion*. International journal of control, 55(3) :531–534, 1992.

[9] McCarthy, John: *Formalizing common sense*. Ablex, Norwood, New Jersey, 1990.

[10] Minsky, Marvin: *A framework for representing knowledge*. 1974.

[11] Minsky, Marvin: *The emotion machine*. New York : Pantheon, 56, 2006.

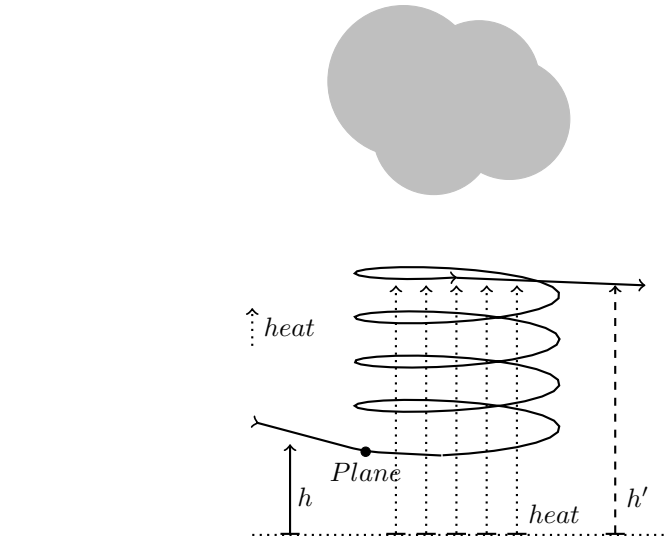
[12] Reiter, Raymond: *A logic for default reasoning*. Artificial intelligence, 13(1-2) :81–132, 1980.

[13] Sion, Maurice: *On general minimax theorems*. Pacific Journal of mathematics, 8(1) :171–176, 1958.

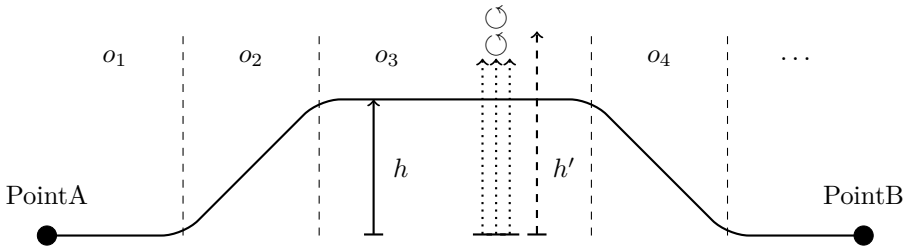
[14] Sitterle, Valerie B., Dane F. Freeman, Simon R. Goerger et Tommer R. Ender: *Systems Engineering Resiliency : Guiding Tradespace Exploration within an Engineered Resilient Systems Context*. Procedia Computer Science, 44 :649 – 658, 2015, ISSN 1877-0509. 2015 Conference on Systems Engineering Research.

[15] Triantaphyllou, Evangelos: *Multi-criteria decision making methods*. Dans *Multi-criteria decision making methods : A comparative study*, pages 5–21. Springer, 2000.

[16] Zhang, Xiaoge, Sankaran Mahadevan, Shankar Sankararaman et Kai Goebel: *Resilience-based network design under uncertainty*. Reliability Engineering & System Safety, 169 :364 – 379, 2018, ISSN 0951-8320.



(a) Thermal phenomena



(b) Possible Scenario

Figure 4.11: Possible path and thermal phenomena.

List of acronyms

ADC

Analog to Digital Converter. 76

AI

Artificial Intelligence. 5, 6, 11, 29–31, 35

BMC

Brushless Motor Controller. 66, 81

DL

Default Logic. 30, 52, 54, 80

FAA

Federal Aviation Agency. 37, 54

FOL

First-Order Logic. 10, 33–38, 46, 48, 80

g

grams. 66

GPS

Global Positioning System. 8, 65, 67, 76, 77, 79

I^2C

Inter-Integrated Circuit. 67, 74, 81

ICSP

In-Circuit Serial Programming –or *In-System Programming*–. 78

IMU

Inertial Measurement Unit. 8, 65, 67, 74, 77

KR

Knowledge Representation. 35, 36

LiPo

Lithium-Polymer. 66, 79, 81

MCU

Microcontroller Unit. 65–69, 77, 78

NML

Non-Monotonic Logic. 36, 54, 80

NMR

Non-Monotonic Reasoning. 29, 30, 32, 35–37, 65

PWM

Pulse-Width Modulation. 9, 55, 69, 70, 77, 79

RMS

Root Mean Square. 55

SPI

Serial Peripheral Interface. 67

UART

Universal Asynchronous Receiver-Transmitter. 67

UAV

Unmanned Aerial Vehicle. 11, 13, 29, 31, 35, 65

V

Volts. 66, 68, 74, 79, 81, 82

W

Watts. 67, 81, 82

ZFC

Zermelo-Fraenkel set. 33

Bibliography

[EER02] El-Azhary, A. Edrees, and A. Rafea. “Diagnostic expert system using non-monotonic reasoning.” In: *Expert Systems with Applications* (2002), pp. 137–144 (cit. on p. 36).

[BKC72] Richard Bellman, BG Kashef, and J Casti. “Differential quadrature: a technique for the rapid solution of nonlinear partial differential equations”. In: *Journal of computational physics* 10.1 (1972), pp. 40–52 (cit. on p. 31).

[BAK02] Dritan Berzati, Bernhard Anrig, and Jürg Kohlas. “Embedding default logic in propositional argumentation systems”. In: *arXiv preprint cs/0207065* (2002) (cit. on p. 38).

[Béz99] Jean-Yves Béziau. “The future of paraconsistent logic”. In: *Logical Studies* 2 (1999), pp. 1–23 (cit. on p. 36).

[Cha10] Ashik Chandra. “Synergy between biology and systems resilience”. MA thesis. MISSOURI UNIVERSITY OF SCIENCE and TECHNOLOGY, 2010 (cit. on p. 50).

[CM96] Todd H Chiles and John F McMackin. “Integrating variable risk preferences, trust, and transaction cost economics”. In: *Academy of management review* 21.1 (1996), pp. 73–99 (cit. on pp. 40, 49).

[CP02] Dragan Cvetkovic and Ian C Parmee. “Preferences and their application in evolutionary multiobjective optimization”. In: *IEEE Transactions on evolutionary computation* 6.1 (2002), pp. 42–57 (cit. on p. 40).

[DGA92] DGAC/SFACT. *Manuel du pilote d’avion*. 5th ed. Ministère des Transports. 1992 (cit. on pp. 15, 23).

[DS15] A. Doncescu and P. Siegel. *DNA Double-Strand Break-Based Non-monotonic Logic*. Computational Biology, Bioinformatics and Systems Biology. Elservier, 2015 (cit. on pp. 45, 48, 71, 79).

[DAK88] Eugene L Duke, Robert F Antoniewicz, and Keith D Krambeer. “Derivation and definition of a linear aircraft model”. In: (1988) (cit. on p. 25).

[Eth87] David W Etherington. “Formalizing nonmonotonic reasoning systems”. In: *Artificial Intelligence* 31.1 (1987), pp. 41–85 (cit. on p. 39).

[FS13] David Fletcher and Mustafa Sarkar. “Psychological resilience: A review and critique of definitions, concepts, and theory.” In: *European Psychologist* 18.1 (2013), p. 12 (cit. on p. 51).

[Fos11] T. Fossen. “MATHEMATICAL MODELS FOR CONTROL OF AIRCRAFT AND SATELLITES.” In: *Department of Engineering Cybernetics, NTNU* (2011) (cit. on p. 31).

[GME14] Simon R. Goerger, Azad M. Madni, and Owen J. Eslinger. “Engineered Resilient Systems: A DoD Perspective”. In: *Procedia Computer Science*. Vol. 28. ELSEVIER, 2014, pp. 865–872 (cit. on p. 50).

[Gui+18] Iain Guilliard, Richard Rogahn, Jim Piavis, et al. “Autonomous Thermalling as a Partially Observable Markov Decision Process (Extended Version)”. In: *arXiv preprint arXiv:1805.09875* (2018) (cit. on p. 29).

[Hol73] Crawford S Holling. “Resilience and stability of ecological systems”. In: *Annual review of ecology and systematics* 4.1 (1973), pp. 1–23 (cit. on pp. 50, 51, 58).

[Hol01] Crawford S Holling. “Understanding the complexity of economic, ecological, and social systems”. In: *Ecosystems* 4.5 (2001), pp. 390–405 (cit. on pp. 51, 53, 57, 80).

[Hol96] Crawford Stanley Holling. “Engineering resilience versus ecological resilience”. In: *Engineering within ecological constraints* 31.1996 (1996), p. 32 (cit. on p. 51).

[Hul07] David G Hull. *Fundamentals of airplane flight mechanics*. Springer, 2007 (cit. on p. 24).

[Kap+17] Ashish Kapoor, Debadeepta Dey, Andrey Kolobov, et al. *Travel path identification based upon statistical relationships between path costs*. US Patent 9,714,831. July 2017 (cit. on p. 30).

[Kol16] Andrey Kolobov. “Integrating Paradigms for Approximate Probabilistic Planning”. In: (2016) (cit. on p. 30).

[LDS13] T. Le, A. Doncescu, and P. Siegel. “Utilization of Default Logic for Analyzing a Metabolic System in Discrete Time.” In: *ICCSA* (2013) (cit. on pp. 45, 48, 79).

[McC90] John McCarthy. *Formalizing common sense*. Norwood, New Jersey: Ablex, 1990 (cit. on p. 35).

[MGN15] Bruce S McEwen, Jason D Gray, and Carla Nasca. “Recognizing resilience: Learning from the effects of stress on the brain”. In: *Neurobiology of stress* 1 (2015), pp. 1–11 (cit. on p. 51).

[MSD18] José Luis Vilchis Medina, Pierre Siegel, and Andrei Doncescu. “Non-monotonie et Resilience: Application au Pilotage d’un Moto-planeur Autonome”. In: (2018) (cit. on p. 85).

[MSD17a] José-Luis Vilchis Medina, Pierre Siegel, and Andrei Doncescu. "Pilotage stable d'un planeur en utilisant une logique non monotone". In: *Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA 2017)*. 2017 (cit. on pp. 73, 84).

[MSD17b] Vilchis Medina, Pierre Siegel, and Andrei Doncescu. "Autonomous Aerial Vehicle Based on Non-Monotonic Logic". In: *3rd International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS)*. 2017, 6p (cit. on p. 84).

[Min74] Marvin Minsky. "A framework for representing knowledge". In: (1974) (cit. on pp. 35, 58, 64).

[Min06] Marvin Minsky. "The emotion machine". In: *New York: Pantheon* 56 (2006) (cit. on pp. 58, 64).

[Moo81] Robert C Moore. "Reasoning about knowledge and action". In: *Readings in Artificial Intelligence*. Elsevier, 1981, pp. 473–477 (cit. on p. 33).

[Moo84] Robert C Moore. *A formal theory of knowledge and action*. Tech. rep. SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1984 (cit. on p. 33).

[Mos04] David Moshman. ": From inference to reasoning: The construction of rationality". In: *Thinking & Reasoning* 10.2 (2004), pp. 221–239 (cit. on p. 40).

[Mur89] Tadao Murata. "Petri nets: Properties, analysis and applications". In: *Proceedings of the IEEE* 77.4 (1989), pp. 541–580 (cit. on p. 31).

[Nan16] A Nancy. "Pilot's handbook of aeronautical knowledge". In: *Washington: US Department of Transportation, Federal Aviation Administration, Flight Standards Service* (2016) (cit. on pp. 15, 29).

[Nie08] Daryle Niedermayer. "An introduction to Bayesian networks and their contemporary applications". In: *Innovations in Bayesian networks*. Springer, 2008, pp. 117–130 (cit. on p. 30).

[PJK12] Jong Jin Park, Collin Johnson, and Benjamin Kuipers. "Robot navigation with MPEPC in dynamic and uncertain environments: From theory to practice". In: *IROS 2012 Workshop on Progress, Challenges and Future Perspectives in Navigation and Manipulation Assistance for Robotic Wheelchairs*. 2012 (cit. on p. 40).

[Rei80] Raymond Reiter. "A logic for default reasoning". In: *Artificial intelligence* 13.1-2 (1980), pp. 81–132 (cit. on pp. 31, 36, 38, 58).

[Rei81] Raymond Reiter. "On closed world data bases". In: *Readings in artificial intelligence*. Elsevier, 1981, pp. 119–140 (cit. on pp. 35, 36, 48).

[SLY17] Meng-Dar Shieh, Yongfeng Li, and Chih-Chieh Yang. "Product Form Design Model Based on Multiobjective Optimization and Multicriteria Decision-Making". In: *Mathematical Problems in Engineering* 2017 (2017) (cit. on p. 40).

[Sio58] Maurice Sion. "On general minimax theorems". In: *Pacific Journal of mathematics* 8.1 (1958), pp. 171–176 (cit. on pp. 40, 49).

[Sit+15] Valerie B. Sitterle, Dane F. Freeman, Simon R. Goerger, et al. "Systems Engineering Resiliency: Guiding Tradespace Exploration within an Engineered Resilient Systems Context". In: *Procedia Computer Science* 44 (2015). 2015 Conference on Systems Engineering Research, pp. 649–658. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.03.013 (cit. on pp. 50, 57).

[TGK18] Samuel Tabor, Iain Guilliard, and Andrey Kolobov. "ArduSoar: an Open-Source Thermalling Controller for Resource-Constrained Autopilots". In: *arXiv preprint arXiv:1802.08215* (2018) (cit. on p. 29).

[Tou11] I. Toulgoat. "Modélisation du comportement humain dans les simulations de combat naval." PhD thesis. Laboratoire SNC/DCNS, 2011 (cit. on p. 45).

[TSD11] I. Toulgoat, P. Siegel, and A. Doncescu. "Modelling of Submarine Navigation by Nonmonotonic Logic." In: *BWCCA* (2011) (cit. on p. 45).

[Tou+09] Isabelle Toulgoat, Julien Botto, Yves De Lassus, et al. "Modeling operator decision in underwater warfare performance simulations". In: *Conference UDT, Cannes., 2009*. 2009 (cit. on p. 45).

[TSL11] Isabelle Toulgoat, Pierre Siegel, and Yves Lacroix. "Operator Behavior Modelling in a Submarine". In: *Combinations of Intelligent Methods and Applications*. Springer, 2011, pp. 21–39 (cit. on p. 45).

[Tri00] Evangelos Triantaphyllou. "Multi-criteria decision making methods". In: *Multi-criteria decision making methods: A comparative study*. Springer, 2000, pp. 5–21 (cit. on pp. 40, 49).

[Vil02] Fabrice Villaumé. "Contribution à la commande des systèmes complexes: application à l'automatisation du pilotage au sol des avions de transport". PhD thesis. Université Paul Sabatier-Toulouse III, 2002 (cit. on p. 26).

[WH17] David D Woods and Erik Hollnagel. "Prologue: resilience engineering concepts". In: *Resilience engineering*. CRC Press, 2017, pp. 13–18 (cit. on p. 51).

[Zha+18] Xiaoge Zhang, Sankaran Mahadevan, Shankar Sankararaman, et al. "Resilience-based network design under uncertainty". In: *Reliability Engineering & System Safety* 169 (2018), pp. 364–379. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2017.09.009 (cit. on pp. 50, 57).

Index