Assuring Safety in an Air Traffic Control System with Defeasible Logic Programming

Sergio Alejandro Gómez[†], Anca Goron[‡], and Adrian Groza[‡]

[†]Artificial Intelligence Research and Development Laboratory (LIDIA)
Department of Computer Science and Engineering
Universidad Nacional del Sur
Av. Alem 1253, (8000) Bahía Blanca, Argentina
Email: sag@cs.uns.edu.ar
[‡]Intelligent Systems Group Department of Computer Science
Technical University of Cluj-Napoca
Baritiu 28, 400391, Cluj-Napoca, Romania
Email: {Anca.Goron,Adrian.Groza}@cs.utcluj.ro

Abstract. Assuring safety in complex technical systems is a crucial issue in several critical applications like air traffic control or medical devices. We present a preliminary framework based on argumentation for assisting flight controllers to reach a decision related to safety constraints in an ever changing environment in which sensor data is gathered at real time.

1 Introduction

Assuring safety in complex technical systems is a crucial issue [1] in several critical applications like air traffic control or medical devices. Air traffic control (ATC) is a service provided by ground-based controllers who direct aircraft on the ground and through controlled airspace, providing advisory services to aircraft in non-controlled airspace. The primary purpose of ATC is to prevent collisions, organize the flow of traffic, and provide information and support for pilots. In this context, accidents are mainly produced by human errors. Such accidents can be avoided by verifying the safety for the ATC system in a logical manner in order to produce support for human air controllers to make rationally justified decisions.

Argumentation [2] provides a sophisticated mechanism for the formalization of common-sense reasoning. Intuitively, an argument can be thought of as a coherent set of statements that supports a claim. The ultimate acceptance of an argument will depend on a dialectical analysis of arguments in favor and against the claim. Defeasible Logic Programming (DeLP) [3] is a reasoning framework based on logic programming and defeasible argumentation with a working implementation.[1]

Safety assurance and compliance to safety standards-based methods of certification such as DO-178B [4] may prove to be a real challenge when having to deal

[1] See http://lidia.cs.uns.edu.ar/delp_client/.

with adaptive systems, in which it is necessary to handle continuous changes. As traditional methods are not very effective in this, argument-based safety cases offer a plausible alternative basis for certification in these fast-moving fields. Our hypothesis is that argumentation can be used to assure safety in complex critical systems by providing a way of assisting end-users to reach rationally justified decisions. In this paper we propose a decision support system for an ATC based on DeLP. Landing criteria for assuring safety in complex landing situations are modeled as a DeLP program. Prospective decisions are presented to the system as queries. Given a query representing a decision concerning a safety requirement w.r.t. such a set of criteria, the DeLP engine will engage in an introspective dialectical process considering pros and cons against a decision and will answer a recommendation in the case that there is a warrant for the query. Besides, as in a real-time environment in which border conditions may vary from second to second, decisions cannot be taken with respect to a static DeLP program. Thus, we present a preliminary framework for making recommendations based on sensor input regarding the values of the parameters characterizing the safety problem.

Outline: In Sect. 2 we present the fundamentals of Defeasible Logic Programming. In Sect. 3 we present a framework for performing safety verification in an ATC system along with a case sudy. In Sect. 4 we discuss how to extend the framework for performing continuous reasoning on sensor feed regarding a safety decision. In Sect. 5 we review related work. Finally, in Sect. 6 we conclude.

2 Defeasible Logic Programming

Defeasible Logic Programming (DeLP) [3] provides a language for knowledge representation and reasoning that uses *defeasible argumentation* to decide between contradictory conclusions through a *dialectical analysis*, and providing a good trade-off between expressiveness and implementability for dealing with incomplete and potentially contradictory information. In a DeLP program $\mathcal{P} = (\Pi, \Delta)$, a set Π of strict rules $P \leftarrow Q_1, \ldots, Q_n$ (which encode certain knowledge), and a set Δ of defeasible rules $P \prec Q_1, \ldots, Q_n$ (which encode knowledge with possible exceptions) can be distinguished. An *argument* $\langle \mathcal{A}, H \rangle$ is a minimal non-contradictory set of ground defeasible clauses $\mathcal{A}$ of Δ that allows to derive a ground literal H possibly using ground rules of Π. Since arguments may be in conflict (concept captured in terms of a logical contradiction), an attack relationship between arguments can be defined. To decide between two conflicting arguments, we will use *generalized specificity* — a syntactic criterion that prefers arguments more informed and arguments based on shorter derivations. If the attacking argument is strictly preferred over the attacked one, then it is called a *proper defeater*. If no comparison is possible, or both arguments are equi-preferred, the attacking argument is called a *blocking defeater*. To determine whether a given argument $\mathcal{A}$ is ultimately undefeated (or *warranted*), a dialectical process is recursively carried out, where defeaters for $\mathcal{A}$, defeaters for these defeaters, and so on, are taken into account. Given a DeLP program $\mathcal{P}$ and a

query H, the final answer to H w.r.t. $\mathcal{P}$ is based on such dialectical analysis. The answer to a query can be: *Yes* (when there exists a warranted argument $\langle \mathcal{A}, H \rangle$), *No* (when there exists a warranted argument $\langle \mathcal{A}, \sim H \rangle$), *Undecided* (when neither $\langle \mathcal{A}, H \rangle$ nor $\langle \mathcal{A}, \sim H \rangle$ are warranted), or *Unknown* (when H does not belong to $\mathcal{P}$).

3 Safety Verification for Air Traffic Control Systems

We now present a safety verification framework for an ATC system based on DeLP.

Definition 1 (Safety verification system). *A* safety verification system $\mathcal{V}$ *is a pair* $(\mathcal{P}, \mathcal{S})$ *where* $\mathcal{P}$ *is a DeLP program establishing logical criteria for assuring safety and* $\mathcal{S}$ *is a set of literals containing sensor information of the environment. The language* $\mathcal{L}_\mathcal{V}$ *of the safety verification system is the set of all literals in* $\mathcal{P} \cup \mathcal{S}$.

Definition 2 (Prospective safety decision). *Let* $\mathcal{V} = (\mathcal{P}, \mathcal{S})$ *be a safety verification system. A* prospective safety decision *is a literal in* $\mathcal{L}_\mathcal{V}$.

Definition 3 (Safety recommendation). *Let* $\mathcal{V} = (\mathcal{P}, \mathcal{S})$ *be a safety verification system and* $\mathcal{D}$ *be a prospective safety decision. A safety recommendation for* $\mathcal{D}$ *is either one of:*

- *Perform: If there is a warranting argument for* $\mathcal{D}$ *w.r.t. the DeLP program* $\mathcal{P} \cup \mathcal{S}$.
- *Do not perform: If there is a warranting argument for* $\sim\mathcal{D}$ *w.r.t. the DeLP program* $\mathcal{P} \cup \mathcal{S}$.
- *Unable to Reach Recommendation: When there is neither a warranted argument for* $\mathcal{D}$ *nor* $\sim\mathcal{D}$ *w.r.t. the DeLP program* $\mathcal{P} \cup \mathcal{S}$.
- *Not Applicable: Whenever* $\mathcal{D}$ *does not belong to* $\mathcal{L}_\mathcal{V}$.

Example 1. In Fig. 1 we present an example of a DeLP program $\mathcal{P}$ for defining a safety verification system for an ATC. The main system safety verification is clearance for landing denoted by the literal $clearance(ID, A, T)$, meaning that flight ID has clearance to land on runway A at time T. The meaning of the safety criteria are as follows: Flight ID usually has clearance to land on runway A at time T if ID is not forbidden to land on runway A at time T and viceversa. A flight has clearance to land at time T if it has had a critical failure T. It is allowed to land at time T on runway A if the wind is calm on runway A at time T and it is forbidden if it is windy at that time. A runway is windy at time T if the wind speed is greater than 15 knots, it is calm otherwise. Exceptionally a flight is allowed to land on a windy runway if it has fuel trouble. A flight is considered to have fuel trouble if its remaining fuel allows it to fly less than 15 minutes, otherwise the flight has no fuel trouble.

In Fig. 2 we present sensor information $\mathcal{S}$ for the rules presented above. There is one runway called *r01* and only one flight called *f701*. We will consider

$clearance(ID, A, T) \prec \sim forbidden(ID, A, T), runway(A), flight(ID).$
$\sim clearance(ID, A, T) \prec forbidden(ID, A, T), runway(A), flight(ID).$
$clearance(ID, A, T) \leftarrow critical\_failure(ID, T).$
$\sim forbidden(ID, A, T) \prec calm(A, T).$
$forbidden(ID, A, T) \prec windy(A, T).$
$\sim forbidden(ID, A, T) \prec fuel\_trouble(ID, T).$
$\sim forbidden(ID, A, T) \prec fuel\_trouble(ID, T), windy(A, T).$
$windy(A, T) \prec wind\_speed(A, S, T), S > 15.$
$calm(A, T) \prec wind\_speed(A, S, T), S < 15.$
$fuel\_trouble(ID, T) \prec remaining\_fuel(ID, R, T), R < 15.$
$\sim fuel\_trouble(ID, T) \prec remaining\_fuel(ID, R, T), R > 15.$

Fig. 1. Logical criteria for assuring safety in an air control system

the safety verification system $(\mathcal{P}, \mathcal{S})$, we will introduce sensor information and show how the proposed approach can recommend a decision to a human traffic controller based on a dialectical analysis performed on $\mathcal{P} \cup \mathcal{S}$.

At time 0, for flight *f701* we only have its identification as there is only a fact *flight*(*f701*) to consider. In this case the safety recommendation for the prospective decision *clearance*(*f701*, *r01*, 0) is *Unable to Reach Recommendation* as DeLP answer for query *clearance*(*f701*, *r01*, 0) is *Undecided* because no argument can be built for that literal from the information available.

At time 10, for flight *f701* we know that the wind speed at runway *r01* is 3 knots. The prospective decision represented by the literal *clearance*(*f701*, *r01*, 10) has *Perform* as safety recommendation because there exists a warranting argument $\mathcal{A}_1$ for it:

$$\mathcal{A}_1 = \left\{ \begin{array}{l} clearance(f701, r01, 10) \prec \sim forbidden(f701, r01, 10), runway(r01), flight(f701) \\ \sim forbidden(f701, r01, 10) \prec calm(r01, 10) \\ calm(r01, 10) \prec wind\_speed(r01, 3, 10), 3 < 15 \end{array} \right\},$$

meaning that flight *f701* has permission to land on runway *r01* because it is a calm runway as the wind speed is only 3 knots at time 10.

When we consider the situation for flight *f701* at time 20, we see that wind speed at runway *r01* is 30 knots (making it very windy). Then the safety recommendation for the prospective decision *clearance*(*f701*, *r01*, 20) is *Do not perform* as there is an argument $\langle \mathcal{A}_2, \sim clearance(f701, r01, 20) \rangle$ where:

$$\mathcal{A}_2 = \left\{ \begin{array}{l} \sim clearance(f701, r01, 20) \prec forbidden(f701, r01, 20), runway(r01), flight(f701) \\ forbidden(f701, r01, 20) \prec windy(r01, 20) \\ windy(r01, 20) \prec wind\_speed(r01, 30, 20), 30 > 15 \end{array} \right\}.$$

The next case shows how having information about fuel reserve comes into play. We see that at time 30, at runway *r01* wind speed is under 16 knots and the flight *f701* has fuel for 60 minutes, which is plenty of time. In this case, in regards to the prospective decision *clearance*(*f701*, *r01*, 30) the safety recommendation of the system is *Do not perform* as an argument similar to the previous situation can be built: $\langle \mathcal{A}_3, \sim clearance(f701, r01, 30) \rangle$ where

$$\mathcal{A}_3 = \left\{ \begin{array}{l} \sim clearance(f701, r01, 30) \prec forbidden(f701, r01, 30), runway(r01), flight(f701) \\ forbidden(f701, r01, 30) \prec windy(r01, 30) \\ windy(r01, 30) \prec wind\_speed(r01, 40, 30), 40 > 15 \end{array} \right\}.$$

Notice that in this case, the information about the remaining fuel is apparently not taken into account but it actually is as we show in the next case.

When we consider the case of flight *f701*, at time unit 40, there is a wind speed of 16 knots at runway one, which is not much but it is over the safety limit of 15 knots. Besides *f701* has only 12 minutes left of fuel. We will see that the answer for the query $clearance(f704, r01, 40)$ is obtained through an interesting dialectical process modeled by the dialectical tree presented in Fig. 3 making the recommendation to be *Perform*. We can see that there is an argument $\langle \mathcal{A}_6, clearance(f701, r01, 40)\rangle$ (expressing that the plane can land because it has fuel trouble having only 12 minutes left) which is defeated by another argument $\langle \mathcal{B}_6, \sim clearance(f701, r01, 40)\rangle$ (saying that the plane cannot land because it is windy) that in turn is defeated by $\langle \mathcal{C}_6, \sim forbidden(f701, r01, 40)\rangle$ (that says that the plane can land whenever it has fuel trouble in windy conditions), thus reinstating $\mathcal{A}_6$; on the other hand $\mathcal{A}_6$ is also defeated by $\langle \mathcal{D}_6, forbidden(f701, r01, 40)\rangle$ (arguing that the plane should be forbidden to land because of windy conditions) which in turn is also defeated by $\mathcal{C}_6$ where:

$$\mathcal{A}_6 = \left\{ \begin{array}{l} clearance(f701, r01, 40) \prec \sim forbidden(f701, r01, 40), runway(r01), flight(f701) \\ \sim forbidden(f701, r01, 40) \prec fuel\_trouble(f701, 40) \\ fuel\_trouble(f701, 40) \prec remaining\_fuel(f701, 12, 40), 12 < 15 \end{array} \right\},$$

$$\mathcal{B}_6 = \left\{ \begin{array}{l} \sim clearance(f701, r01, 40) \prec forbidden(f701, r01, 40), runway(r01), flight(f701) \\ forbidden(f701, r01, 40) \prec windy(r01, 30) \\ windy(r01, 40) \prec wind\_speed(r01, 16, 40), 16 > 15 \end{array} \right\},$$

$$\mathcal{C}_6 = \left\{ \begin{array}{l} \sim forbidden(f701, r01, 40) \prec fuel\_trouble(f701, 40), windy(r01, 40) \\ fuel\_trouble(f701, 40) \prec remaining\_fuel(f701, 12, 40), 12 < 15 \\ windy(r01, 40) \prec wind\_speed(r01, 16, 40), 16 > 15 \end{array} \right\} \text{and}$$

$$\mathcal{D}_6 = \left\{ \begin{array}{l} forbidden(f701, r01, 40) \prec windy(r01, 40) \\ windy(r01, 40) \prec wind\_speed(r01, 16, 40), 16 > 15 \end{array} \right\}.$$

At time unit 50, we see that when there is a critical failure on the plane, the system will allow it to land independently of runway conditions. So the safety recommendation for the prospective decision $clearance(f701, r01, 50)$ will be *Perform* as a warranted argument $\langle \emptyset, clearance(f701, r01, 50)\rangle$ can be found, which is based on the derivation using the strict rule $clearance(f701, r01, 50)) \leftarrow critical\_failure(f701, 50)$.

Notice that at time unit 60, a richer situation arises: In this scenario the flight has a critical malfunction at time 60 and sensor information indicates that wind speed at runway *r01* is also 100 knots at that time. The safety recommendation for the safety prospective decision $clearance(f701, r01, 60)$ is *Perform* based on the evidence that the DeLP answer for $clearance(f701, r01, 60)$ is *Yes* as there is an empty warranting argument. However notice that in this case the DeLP answer for $windy(r01, 60)$ is *Yes* and the answer for $forbidden(f701, r01, 60)$ is also *Yes*, but in this case the argument for $\sim clearance(f701, r01, 60)$ based on the *defeasible* rule $\sim clearance(f701, r01, 60) \prec forbidden(f701, r01, 60)$ is not activated as according to the specificity criterion for comparing arguments it is weaker than the argument for $clearance(f701, r01, 60)$ based on the *strict* rule $clearance(f701, r01, 60) \leftarrow critical\_failure(f701, 60)$, so the latter is preferred.

$runway(r01)$.
$wind\_speed(r01, 3, 10)$.
$wind\_speed(r01, 40, 30)$.
$wind\_speed(r01, 16, 40)$.
$critical\_failure(f701, 50)$.
$critical\_failure(f701, 60)$.
$flight(f701)$.
$wind\_speed(r01, 30, 20)$.
$remaining\_fuel(f701, 60, 30)$.
$remaining\_fuel(f701, 12, 40)$.
$wind\_speed(r01, 100, 60)$.

Fig. 2. Sensor information for the air control system

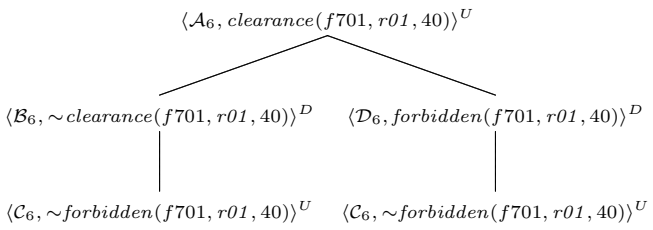

Fig. 3. Dialectical tree for the query $clearance(f701, r01, 40)$

4 Continuous Reasoning with DeLP

We now extend the framework presented above for producing recommendations to assist in deciding about critical safety issues of an air traffic control system. The scenario presented in Ex. 1 showed that DeLP can be used to characterize a recommender system for assisting human air-traffic controllers. However, in a real-time environment where border conditions do not remain fixed as implied by the use of an static DeLP program, the framework presented in Sect. 3 is clearly insufficient. Because of this, we now extend it for including sensor information that continuously feeds the DeLP system. In this regard, we show how the DeLP system can produce a stream of recommendations in real time according to the input fed to the system.

Definition 4 (Data stream). *A* data stream $S(key, \tau) = \langle v_1, v_2, \ldots, v_i, \ldots\rangle$ *is an infinite list of values* v_i *for the given key updated at time step* τ*. When a sensor fails to collect a sample measure at a certain time unit, the value* $\perp$ *is added to the data stream.*

Example 2. In the scenario defined by the DeLP program $\mathcal{P}$ in Ex. 1, we can consider that there are three streams of sensor data: $S_{wind\_speed}(r01, 10)$, $S_{remaining\_fuel}$ $(f701, 10)$ and $S_{critical\_failure}(f701, 10)$. The first stream provides the wind speed at runway $r01$ at each time unit, the second stream updates the remaining fuel for the flight $f701$ also at each time unit, and the third informs about a critical failure situation in flight $f701$. In the running example, the stream $S_{wind\_speed}(r01, 1) = \langle 3, 30, 40, 16, \perp, 100\rangle$ will generate the DeLP facts presented in Fig. 2, namely $wind\_speed(r01, 3, 10)$, $wind\_speed(r01, 30, 20)$, $wind\_speed(r01, 40,$ $30)$, $wind\_speed(r01,$ $16, 40)$ and $wind\_speed(r01, 100, 60)$.

As according to Fig. 2, we only have information for the remaining fuel at time units 30 and 40, the stream $S_{remaining\_fuel}(f701, 10) = \langle\perp, \perp, 60, 12, \perp, \perp\rangle$ is

stored as the DeLP facts: *remaining_fuel*($f701, 60, 30$) and *remaining_fuel*($f701, 12$, 40). Regarding the critical failure sensor at flight $f701$, the stream $S_{critical\_failure}$ ($f701$, 10) $= \langle \bot, \bot, \bot, \bot, yes, yes \rangle$ will produce the facts: *critical_failure*($f701, 50$) and *critical_failure*($f701, 60$).

A continuous query is a query executed continuously using data arriving from sensors. Formally:

Definition 5 (Continuous query). *A* continuous query $CQ_{\mathcal{P}} = (Q, \mathcal{S}, \tau)$, *is a query Q posed to a DeLP program $\mathcal{P}$ executed continuously at a specified time-step τ against data arriving from a set $\mathcal{S}$ of sensor streams.*

Example 3. The continuous query

$$CQ_{\mathcal{P}} = (clearance, \{S_{wind\_speed}(r01, 10), S_{remaining\_fuel}(f701, 10), S_{critical\_failure}(f701, 10)\}, 1)$$

investigates the validity of the *clearance* predicate at each time unit, based on the wind speed at runway $r01$, the remaining fuel of flight $f701$ and the critical failure sensor at flight $f701$. At time instance $\tau = 1$, the query is computed based on the data that arrive at each time step from the two input streams.

A continuous query generates an output stream of safety decisions, which are computed based on the semantics of DeLP.

Definition 6 (Chain of safety recommendations). *A* chain of safety recommendations $S_{\mathcal{D}}(p_1, \ldots, p_n, \tau)$ *is a stream of $\mathcal{D}$ prospective safety decisions about parameters $p_1, \ldots, p_n$ starting at time unit τ and annotated with the time-step in which $\mathcal{D}$ is valid.*

Example 4. Recalling the answers of the system presented at Ex. 1, the chain of safety recommendations would be:

$$\begin{aligned} S_{clearance}(f701, r01, 0) = \langle &(\textit{Unable to Reach Recommendation}, 0), (\textit{Perform}, 10), \\ &(\textit{Do not perform}, 20), (\textit{Do not perform}, 30) \\ &(\textit{Perform}, 40), (\textit{Perform}, 50), (\textit{Perform}, 60) \rangle. \end{aligned}$$

5 Related Work

Nakamatsu *et al.* [5] propose a theoretical framework for a logical safety verification for an air traffic control system based on a paraconsistent logic program called an Extended Vector Annotated Logic Program with Strong Negation (EVALPSN for short). EVALPSN uses numerical weights for assigning credibility to facts, as our approach relies on DeLP and on argument construction and a dialectical process for determining which conclusions, our solution relieves the knowledge engineer of the burden of weighing his information. Capobianco *et al.* [6] define ODeLP which allows to use DeLP in a dynamic environment based on facts obtained through *observations*. ODeLP allows to precompile the status of arguments in order to compute warrants more efficiently. Our approach could benefit from ODeLP in the case of programs formed only by defeasible rules because ODeLP does not accept strict rules as needed by our proposal.

Goron *et al.* [7] present an application of argumentation for supporting autonomous decision making performed by an unmanned aerial vehicle (UAV), extending its use for hybrid logics model update. Argumentation theory is considered for assisting the process of updating a Kripke model, which is viewed as a snapshot of the world that they are interested in. When the model fails to verify a property, a DeLP program is run to analyze the current state and deciding on an update operation of the model. Unlike our solution, a Kripke structure is used to capture the operation of the UAV, and DeLP is applied in repairing the initial model such as to comply to safety regulations. In our case, DeLP is used to assist a human controller.

6 Conclusions and Future Work

We presented a preliminary framework for implementing a recommender system for an air traffic control used for assisting flight controllers to reach rational decisions related to a safety constraint. We also consider how to adapt DeLP to an ever changing environment in which information sensor is gathered at real time, with a running scenario. Much work remains to be done such as exploring properties of the approach along with the feasibility of implementation as a real-world application. In that regard, infinite lists can be processed with a functional programming approach based on lazy evaluation as suggested by [8].

Acknowledgments: Part of this work was supported by the Argentina-Romania Bilateral Agreement entitled "ARGSAFE: Using argumentation for justifying safeness in complex technical systems" (MINCYT-MECTS Project RO/12/05) and Universidad Nacional del Sur, Argentina. Adrian Groza is supported by the intern research project at Technical University of Cluj-Napoca, Romania: "GREEN-VANETS: Improving transportation using Car-2-X communication and multi agent systems".

References

1. Graydon, P., Kelly, T.P.: Using argumentation to evaluate software assurance standards. Information and Software Technology **55**(9) (2013) 1551–1562
2. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artificial Intelligence **171**(10-15) (2007) 619–641
3. García, A., Simari, G.: Defeasible Logic Programming an Argumentative Approach. Theory and Practice of Logic Programming **4**(1) (2004) 95–138
4. Rushby, J.: A safety-case approach for certifying adaptive systems. In: In AIAA Infotech@Aerospace Conference, American Institute of Aeronautics and Astronautics, John Rushby. (2009)
5. Nakamatsu, K., Suito, H., Abe, J., Suzuki, A.: Paraconsistent logic program based safety verification for air traffic control. In: 2002 IEEE International Conference on Systems, Man and Cybernetics. (2002)
6. Capobianco, M., Chesñevar, C., Simari, G.: An argument-based framework to model an agent's beliefs in a dynamic environment. In: AAMAS 2004. (July 2004) 163–178
7. Goron, A., Groza, A., Gómez, S.A., Letia, I.A.: Towards an argumentative approach for repair of hybrid logics models. In: ArgMAS 2014. (2014) (Accepted)
8. Groza, A., Letia, I.: Plausible description logics programs for stream reasoning. Future Internet (4) (2012) 865–881