Flight Procedures Description Using Semantic Roles

Cédric Tarbouriech, Denys Bernard, Laure Vieu, Adrien Barton,
Jean-François Éthier

Flight Procedures Description Using Semantic Roles

Cédric Tarbouriech[1], Denys Bernard[2], Laure Vieu[1,3], Adrien Barton[1,4] and Jean-François Éthier[4]

[1]*Institut de Recherche en Informatique de Toulouse (IRIT), Université de Toulouse & CNRS, France*
[2]*Airbus, Architecture & Integration IIVD, Blagnac, France*
[3]*Laboratorio di Ontologia Applicata, ISTC-CNR, Italy*
[4]*Groupe de Recherche Interdisciplinaire en Informatique de la Santé (GRIIS), Sherbrooke University, Québec, Canada*

Abstract

This paper presents an ontological analysis of flight procedures. We aim at representing participants of processes described by such procedures and what roles they play. We re-use the Process Specification Language (PSL) by re-interpreting it as dealing with informational entities describing processes rather than process universals. The identification of roles is based on the analysis of flight manuals (FCOM). The instructions were analysed to find out patterns that supported the construction of an ontology of instructions.

Keywords

flight procedure, semantic roles, FCOM, PSL, procedure

1. Introduction

For a virtual assistant to be able to help a person execute some process, it is necessary to have a representation of the sequence of steps involved and the participants (living beings, objects, informational entities, etc.) involved, among other entities. To do so, the virtual assistant can be based either on a representation of a particular of procedure (a procedure is an informational entity, e.g. the text of a chocolate cake recipe) or on a representation of an universal of executions of this procedure (an execution is a process, e.g. baking a chocolate cake following the recipe). Here, we chose to use the procedure particular because of the challenges that would be raised by the other solutions. Indeed, using universals would expose us to a common problem of structural universals (see Section 3.1.1). Moreover, two particular executions of the same procedure are likely to differ to some extent (they may even not follow appropriately some parts of the procedure): they are typically realised at different times, involving different particular participants, and under different conditions. For these reasons, the present work will focus on the representation of procedures.

A procedure presents various features that need to be represented, and refer to: the temporal structure of the directed actions (the order in which the actions must be done); the participants

FOMI 2021: 11th International Workshop on Formal Ontologies meet Industry, held at JOWO 2021: Episode VII The Bolzano Summer of Knowledge, September 11–18, 2021, Bolzano, Italy
cedric.tarbouriech@irit.fr (C. Tarbouriech); denys.bernard@airbus.com (D. Bernard); laure.vieu@irit.fr (L. Vieu)
https://www.irit.fr/~Cedric.Tarbouriech/ (C. Tarbouriech); https://www.irit.fr/~Laure.Vieu/ (L. Vieu)
0000-0001-8119-7826 (C. Tarbouriech); 0000-0002-0131-4795 (D. Bernard); 0000-0003-0303-0531 (L. Vieu); 0000-0001-5500-6539 (A. Barton); 0000-0001-9408-0109 (J. Éthier)

(the endurants involved in the action: human beings, objects, etc.); the conditional structure (some actions must be done only under some circumstances); the mereological structure (some complex actions may be decomposed into more basic ones), etc.

This paper presents a work inspired by PSL (Process Specification Language) [1], a framework to describe the structure of universals of executions. Even though the work on which it is based targeted multiple features of procedures, this paper focuses on the integration of participants in the representation. PSL uses universals of processes (called "activities") and particulars of processes (called "activity occurrences"). In this paper, the activities are reinterpreted as informational entities describing such processes, and thus as endurants rather than universals of perdurants. We also extend PSL to represent a variety of participants[1] by introducing semantic roles, which are relations between activities and participants.

The proposed ontology[2] is based on a corpus created for this work, that contains procedures from flying procedure manuals named "FCOMs" (The acronym FCOM stands for *Flight Crew Operating Manual*). The resulting representation framework is meant to be used in applications such as in-flight virtual pilot assistants.

2. Methods

The first step of the integration of semantic roles into PSL was to get a good overview of what kinds of roles are used in instructions, and under which modalities. Therefore, the first task was the creation of a corpus of instructions, based on FCOMs. This corpus was then analysed with the aim of identifying patterns of instructions, where a pattern is the combination of roles involved in an instruction. Instructions sharing the same pattern were then clustered to make a taxonomy of the piloting activities. In this section, we describe how the corpus was built and present PSL on which our proposal builds.

2.1. Creation of the Corpus

2.1.1. Presentation of FCOMs

FCOMs are manuals of all the procedures that pilots may need, whether during regular situations (e.g. take-off and landing procedures) or emergencies. There are various FCOMs, depending on the airplane or the company. The work presented in this paper uses all the procedures of a generic A350 FCOM [2]. The procedure partly presented in Figure 1 is a part of the Take-off procedure, namely the Thrust Setting sub-procedure, and it will be used as an example in the paper.

Each (elementary) instruction line has at least two parts, separated by a row of dots. A synthetic representation of an instruction is presented in Figure 2. The challenge is an endurant involved in the action, such as a tool (THRUST LEVER), a person or a group of people (CREW) or a message ("TAKE-OFF"). The *response* part is an indication of what to do on the object. It may

[1] Since activities are reinterpreted as informational entities, the participants participate in the processes described by those activities, and not in those activities.

[2] The ontology is available on the following Github repository: https://github.com/CedricTarbouriech/FPO.

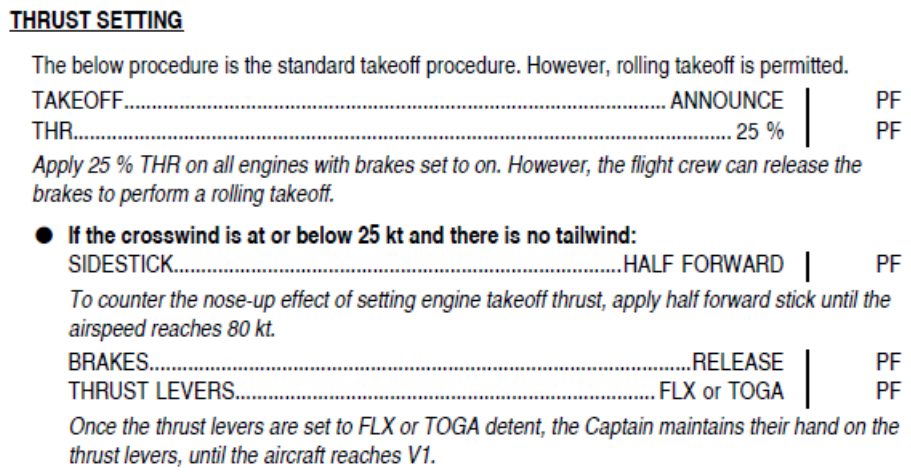

Figure 1: Excerpt of the A350 Take-Off procedure (image lightly edited to remove irrelevant information).

be an action (ANNOUNCE) or a state (HALF FORWARD). The *agent*[3] gives an indication about who is supposed to do the action. It may designate a single person (PF[4]) or a group (BOTH PILOTS). Some instructions have a free-text paragraph, the *comment*, which indicates important information about the instruction execution. The *agent* or the *comment* may be missing. The first instruction in Figure 1 is therefore understandable as "Agent PF must announce the take-off".

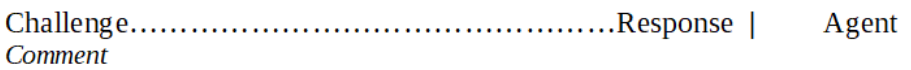

Figure 2: Representation of an instruction

The temporal structure of the procedure is usually indicated by the order of the instructions. However, some instructions can have annotations to indicate that they can be executed at any time. The conditional structure is expressed using bullets.[5] All the conditioned instructions are indented under the condition label. A conditional block is visible in Figure 1: the bullet is the condition, all the lines below it are conditioned by it. The mereological structure is based on redirection to other common procedures (not illustrated in Figure 1). These common procedures are written as appendices and may be referenced anywhere in the FCOM.

2.1.2. Instructions Extraction

FCOMs are XML-based documents, i.e. the manuals are generated from the procedures structured in XML. Besides XML offering a certain level of structuration (as shown in Figure 3, the

[3]Note that "participant" and "agent" refer to two different things: the former refers to any entity that is involved in the action, whereas the latter always refers to the entity performing the action. The agent is a kind of participant.

[4]"PF" means "Pilot Flying". The other pilot is called "PM", for "Pilot Monitoring".

[5]Different bullet styles are used to indicate exclusive or non-exclusive conditions.

```
<action lid="P.00006607.0001001.008" crm-fc="PF">
  <cr-action code="00006607.0001001.004">
    <challenge>TAKEOFF</challenge>
    <response>ANNOUNCE</response>
  </cr-action>
</action>
```

Figure 3: The first instruction of the Figure 1 encoded in the XML file (code lightly edited to remove irrelevant information).

instruction parts are separated using different XML elements), the various parts of instructions (*challenge*, *response*, etc.) are mainly written in plain text. Therefore, there are small variations requiring a normalisation of the corpus.

To have a usable corpus, the parts of the instructions have to be extracted. Among the four parts presented in Figure 2, only the *challenge* and the *response* were extracted. The *comment* was left apart because, in contrast to the other elements, it is not just constituted by a word or two, but by one or more sentences. Thus, extracting information from it would require Natural Language Processing techniques that were not in the scope of this work. However, in rare cases, comments were used to better understand instructions. The *agent* was also left apart. Procedures are executed by human beings, hence we consider that every instruction has at least one agent. If the instruction involves multiple persons, the agent is considered to be the group of those persons. So, there is at most one agent. Therefore, for each instruction, there is exactly one agent. As there is such a role for every instruction, we did not extract the *agent* parts from the FCOM. The two extracted elements were grouped and will be called a "pair".

The normalisation consisted mainly of typographic modifications. Indeed, there were some typographic variations, such as upper or lower case being used in similar contexts, or the inclusion of hyphens or spaces. Those normalisations were made semi-automatically. The data were explored manually. Many instructions were rewritten under standardised forms, which were chosen among the variations, mainly by selecting the most frequent one. 1485 unique pairs of challenge and response were extracted.

Once the normalisation process was performed, the corpus was reduced by selecting some pairs. Two different sets of pairs were selected. The first selection set contains all the pairs that appear five times or more in the document, i.e. 144 pairs (approximately 10% of the corpus). It ensures that the remaining pairs were highly representative of what could be found in the FCOM. The second set contains also 144 randomly selected pairs among the remaining pairs. It ensures that the pairs considered are diverse enough. To ensure higher diversity, a condition was added: to be selected in the second set, a pair must have a response part that is not in the first extracted response set. Finally, once both sets are merged, the post-selection corpus contains 288 pairs, which is approximately 20% of the 1485 pairs of the pre-selection corpus.

The corpus was analysed to identify patterns and involved roles. Results are exposed in 3.1.

2.2. Presentation of PSL

The Process Specification Language ontology provides a description framework for manufacturing processes. It is described by the ISO standard 18629 [3]. PSL has a semantics based on first-order logic and an OWL and SWRL axiomatisation.[6] The version of PSL used for this work is described in Grüninger's 2009 paper [1]. PSL proposes core theories and definitional extensions. The first core theory, which is extended by other core theories, is PSL-Core. Even though the other core theories are consistent with PSL-Core, they may be mutually inconsistent. The core theories used in this work are T_{psl_core} and $T_{subactivity}$.

The T_{psl_core} theory defines the basic classes and relations used in all the PSL environment, presented in Table 1. This theory defines four classes, all disjoint, two relations and two functions, three of them being time-related. Within PSL, each activity is a reified universal that is an instance of *activity*. Temporal relations only apply to occurrences.

Table 1
Vocabulary of T_{psl_core} [1].

$activity(a)$	a is an activity
$activity\_occurrence(o)$	o is an activity occurrence
$timepoint(t)$	t is a timepoint
$object(x)$	x is an object
$occurrence\_of(o,a)$	o is an occurrence of a
$beginof(o)$	the beginning timepoint of o
$endof(o)$	the ending timepoint of o
$before(t_1,t_2)$	timepoint t_1 precedes timepoint t_2 on the timeline

The $T_{subactivity}$ theory introduces the class *Primitive*, subclass of *Activity*, and the relation *subactivity* (see Table 2). These two are used to define a mereological structure over activities, in which *Primitive* instances are the atomic elements. The axioms of the theory make *subactivity* a discrete partial ordering. Except for the discrete feature of the ordering, this is the most basic mereology possible. This theory does not allow the characterisation of the relationship between the occurrence of a complex activity and occurrences of its subactivities. To do so, the core theory $T_{complex}$ is needed.

Table 2
Vocabulary of $T_{subactivity}$ [1].

$subactivity(a_1,a_2)$	a_1 is a subactivity of a_2
$primitive(a)$	a is a minimal element of the *subactivity* ordering

Another theory named T_{actors} was introduced by Katsumi and Grüninger in 2015 [4]. This core theory introduced actors and relations between actors, activities and activity occurrences. The relation *performs* between actors and activities is intended to correspond to the semantic role agent (see below); no other semantic role is considered. However, as far as we know, this theory is only cited in the 2015 paper and never exploited after that. Therefore, we ignored it for this

[6] CLIF and OWL files are accessible at https://github.com/gruninger/colore/tree/master/ontologies.

work. It is worth noting that this work is complementary to ours, as it creates relation between activities and activity occurrences.

3. Results

3.1. Analysis of the Corpus

3.1.1. Instructions

We assume that for the same kind of instructions, the same roles are involved. Therefore, the pairs were grouped by the *response*, which is generally a verb. A class of instructions was associated to each cluster, and named using a verb, as shown in Table 3. This verb is not necessary the *response*. Indeed, sometimes, it was relevant to group clusters using a more generic verb. When such a clustering was done, the classes associated to the clusters were related by subclass relation. For example, the clusters of pairs for which the *response* were "Alert" or "Announce" were grouped under the label "Communicate", and the classes *Alert* and *Announce* are subclasses of *Communicate*.

Table 3
Description of classes based on clusters

Identifier	Description	Examples
Act[7]	Physically act on something. Various subclasses exist, such as: Push, Pull, Press.	"Increase thrust.", "Start chronometer."
Fetch	Fetch a parameter's value from a sensor or another agent. Two subclasses exist: Check and Monitor	"Check if oxygen masks are correctly stowed.", "Monitor fuel consumption."
Communicate	Give information to a human or a group of humans. Various subclasses exist, as: Alert, Order, Announce.	"Announce take-off to passengers.", "Order brace for impact."
Set[7]	Make sure that an object is in a given state.	"Set the thrust levers to 25%.", "Retract flaps."

By a first non-systematic exploration of the corpus, we identified two types of instructions: elementary instructions and meta-instructions. An elementary instruction is an instruction stating that an agent must do some action. A meta-instruction is an instruction that states something about another instruction or group of instructions, such as modifying the deontic structure (e.g., allowing or recommending some action: "Consider landing"), or defining a specific protocol that must be followed each time an action of a given type will be executed. Their range vary, from a

[7]Even though Act and Set look similar, Set is not a subclass of Act. Act is a class of instructions that aim at directing physical actions (push, pull, etc.) on objects (button, lever, etc.), whatever the state of these objects is. Set is a class of instructions that aim at directing actions on objects setting them in some specific states. If the object is already in that state, then no action is performed (except checking the object's state). We could consider Set to be the sum of a Fetch-type action and an Act-type action, conditionally applied depending on the result of the Fetch-type action. However, we used Set as it is considered a simple enough instruction in the FCOM.

single instruction to all subsequent instructions in the procedure. In this work, meta-instructions were not analysed, and therefore various clusters that contain meta-instructions were left out of the analysis, such as "Consider landing" or "For directional control, use rudder". Representing such meta-instructions requires a more complex representation of instructions. Indeed, most instructions are implicitly considered to be orders, and therefore *must* be executed by the pilots; but in "consider landing", the verb "consider" shows that the pilot could, after deliberation, choose to ignore it. "For directional control, use rudder" establishes a way to proceed for precise directional control tasks. It is not an instruction executed once, but it applies to all following instructions that concern directional control.

Let's take an example: consider the procedure "1) Press the button, 2) Flip the lever, 3) Press the button". This procedure has three instructions: 1) and 3) are two utterances of the very same sentence (sequence of words), and 2) is an utterance of another sentence. In that case, PSL will use two particulars, instances of *Primitive*: *pressButton* and *flipLever*. The repetition of "Press the button" will be expressed with activity occurrences linked to *pressButton* with the relation *occurrence_of*. What we want to do is to describe this repetition at the activity level instead of the occurrence level. For example, we want to use activities in temporal relationship. However, if we do this in PSL, the very same particular will be involved in multiple relations. Those relations could be incompatible with each other: stating that *pressButton* is temporally before and after *flipLever* makes no sense. This problem is one of the problems encountered while using structural universals.[8] To avoid it, we want each instance of *Activity* to represent a unique instruction. In this work, each utterance is a particular, and each sentence is associated to one of the subclasses of *Primitive*. In our example, a class *Press*, subclass of *Primitive*, would exist and have both utterances 1) and 3) as instances. To summarize, in PSL, activities are reified universals of processes, but in this paper, they are particulars of informational entities. This is a shift in the interpretation of PSL.

3.1.2. Semantic Roles

In this work, patterns of semantic roles are important to create clusters of instructions. Grouping instructions by role patterns enables to classify more precisely the instructions. The participants (other than the *agent*) were mainly mentioned in the *challenge* part. It is worth noting that a *challenge* could mention multiple participants (e.g. "Advise cabin crew and passengers.") and that some of the participants were not mentioned in the instruction, but in the comment.

To represent the participation of an entity in an action, semantic roles were used. For example, in the FCOM instruction "Agent PF must announce the take-off", "PF" refers to the entity doing (or which will do) the action, namely the agent, and "the take-off" refers to the message (that the take-off is about to occur) to be communicated during this action, namely the theme. There has been various proposals on roles descriptions. We based our work on the ISO norm 24617-4:2014 [10], which defines semantic roles, as presented in Table 4. Among the large variety of roles, only few were chosen, based on their relevance for our problem.

We completed those semantic roles with the *EndLoc* role from the work of Jezek et al. (2014) on Senso Commune [11]. We then redefine the roles to make them better match our needs. This

[8] About structural universals, see Lewis (1986) [5], Armstrong (1986) [6], Bigelow (1986) [7], Fisher (2018) [8] and Garbacz (2020) [9].

Table 4
Relevant semantic roles from ISO norm 24617-4:2014

Role	Description
Agent	Participant in an event who intentionally or consciously initiates an event, and who exists independently of the event.
Goal	Participant in an event that is the (non-locative, non-temporal) end point of an action; the participant exists independently of the event.
Patient	Participant in an event that undergoes a change of state, location, or condition, is causally involved or directly affected by other participants, and exists independently of the event.
Source	Non-locative, non-temporal starting point of an event. The source exists independently of the event.
Theme	Participant in a state or an event that (a) in the case of an event, is essential to the event taking place, but does not have control over the way the event occurs and is not structurally changed by the event, and (b) in the case of a state, is characterized as being in a certain position or condition throughout the state, and is essential to the state being in effect but not as central to the state as a participant in a pivot role. The theme of a state or event exists independently of the state or event.

selection is presented in Table 5. Using these roles, we determined common parts of the clusters' members. They are presented in Table 6.

Table 5
Definition on roles based on our needs

Role	Description	Example
Agent	Participant who initiates the action.	Crew
EndLoc	Final state in which the patient of the action (e.g. a tool) is set.	Full thrust
Goal	Participant that is the recipient of the message communicated by the action.	Passengers
Patient	Participant that undergoes a change of state due to the action.	Thrust levers
Source	Entity from which an information is fetched.	Speed sensor
Theme	Message content transmitted in a communication.	"Prepare for landing", "Take-Off"

3.2. Ontology

The PSL ontology was extended by introducing subclasses of *Primitive*. Each of those subclasses appears in axioms characterising their common pattern exposed in Table 6. This ontology was implemented in OWL.

Table 6
Classes and their roles

Class	Participant 1	Participant 2	Participant 3
Act	Agent	Patient	—
Fetch	Agent	Source	Theme
Communicate	Agent	Goal	Theme
Set	Agent	EndLoc	Patient

3.2.1. Extending the Ontology

Each class representing elementary instructions is a subclass of *Primitive*. The hierarchy is described in Figure 4-a. Each instruction written in the FCOM is an instance of a class. Semantic roles were added in the ontology using properties (in OWL sense), under the format *hasXXX* where *XXX* is a semantic role. All the properties, except for *hasEndloc*, are object properties and have an inverse property, called *isXXXOf*. The *hasEndLoc* property is a data property, whose values are either literals or strings. This is due to the fact that the *EndLoc* role handles tool states. Subclasses were also added to the PSL class *Object*. They are the classes of the different participants. The hierarchy is described in Figure 4-b. *DisplayDevice* is the class of tools that display some value, such as a clock. *Sensor* is the class of tools that record values about the airplane, such as an altimeter. *Trigger* is the class of tools that agents can manipulate, such as levers or buttons. With these roles and classes, we can represent the instruction "PF must set the thrust to 25%." as an instance of the *Set* class. Its formalisation in DL is represented as follows:

Set(instr) *Person*(pf) *Lever*(thrust_lever)
hasAgent(instr, pf) *hasPatient*(instr, thrust_lever) *hasEndloc*(instr, "25%")

Each class of instructions has several axioms about which roles its instances should have. These axioms are based on Table 6, which presents the pattern of roles corresponding to each class of instructions. Note that each instruction particular is in relation with only one participant playing this role. For example, the class *Set* is characterised by DL axioms 1, 2 and 3.

Axiom 1 (Each *Set* has exactly one agent)**.** $Set \sqsubseteq =1 hasAgent.\top$

Axiom 2 (Each *Set* has exactly one patient)**.** $Set \sqsubseteq =1 hasPatient.\top$

Axiom 3 (Each *Set* has exactly one endloc)**.** $Set \sqsubseteq =1 hasEndloc.\top$

A new relation *isComposedOf*, and its inverse *belongsToGroup* (defined by definition 1), are added to represent how groups are composed by their members. Axioms 4 and 5 define the range of the properties.

Axiom 4 (Domain of *isComposedOf*)**.** $\exists isComposedOf.\top \sqsubseteq PersonGroup$

Axiom 5 (Range of *isComposedOf*)**.** $\top \sqsubseteq \forall isComposedOf.Person$

Definition 1 (Definition of *belongsToGroup*)**.** $belongsToGroup \equiv isComposedOf^{-}$

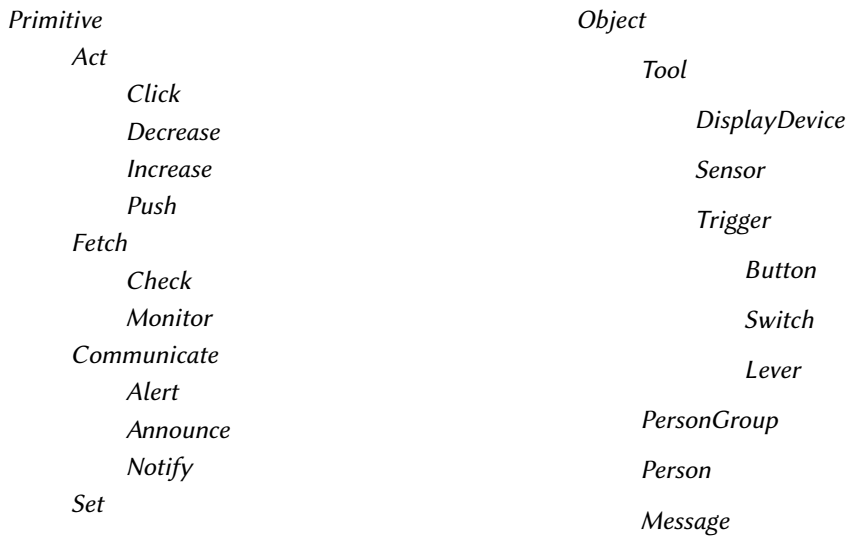

Figure 4: *Primitive* class hierarchy (a) and *Object* class hierarchy (b).

3.2.2. Evaluating the Ontology

The ontology was implemented using Protégé, which helps check the consistency of the ontology. Two procedures from the FCOM, each containing twenty or so instructions, were described and the expressivity of the ontology was tested using SPARQL requests. The SPARQL requests were based on competency questions, such as "Who has to do this action?" or "In which state this tool must be set?".

4. Discussion

4.1. Reduce Instructions Ambiguity for a Virtual Assistant

The ambiguity of instructions at meaning extraction is partially due to the lack of preciseness of the XML schema used. This problem is visible with the *Challenge-Response* pairs. The type of entity the *challenge* part should contain is not clearly determined. For example, *Set* instructions can contain either the name of an entity playing an *EndLoc* role or a *Patient* role. In some rare cases, one of the role players is in the *Response*. It is also important to note that sometimes, one of the role players can be mentioned in the comment, or not at all. Indeed, the executions of some of those instructions rely on the background knowledge of the pilots. This kind of implicit knowledge hinders automatic extraction of the procedures. Defining a clearer framework to write the FCOMs could be a consequent enhancement to achieve automatic extraction. The present work can help to define this framework.

4.2. Tools States Representation

We represented tool states by strings. However, this method is not ambiguity-free. As we already saw, there may be typographic variations between strings. A more general method would be to have an ontology of tools, including sensors, of a plane. This ontology could describe in particular the possible states these tools can be set on.

4.3. Shallow and Deep Roles Representation

Even if the current framework admits various roles, the difference between them is not necessarly clear. The current axiomatisation is shallow and does not differenciate roles other than using their domain and range. For instance, nothing differenciates the roles *agent* and *patient*. A deeper and more complex axiomatisation is needed to have a better grasp of those differences.

4.4. Aligning the Ontology with an Upper Ontology

The change of interpretation of the PSL activities may impede the reuse of the PSL top-level in the current taxonomy. Indeed, in PSL, *Activity* and *Object* were disjoint classes, as the former is a class of universals of perdurants whereas the latter is a class of endurant particulars. However, by classifying activities as informational entities, they arguably are endurants. Therefore the top-level of the taxonomy should be re-examined. This could be done in the light of other upper ontologies, such as BFO [12], DOLCE [13], GFO [14] or UFO [15].

5. Conclusion

This work led to the definition of a framework able to describe roles in procedures, and compatible with the PSL ontology. This framework is a part of a bigger framework developed for a virtual assistant capable of seconding a human agent in the execution of flight procedures.

The presented work allows the description of instructions, i.e. utterances describing actions involving various participants which can be clearly labelled. A taxonomy of the identified actions is also given. Another benefit of this work is the identification of some lack of expressivity in the FCOMs, leading to ambiguity in the instructions and a more complex extraction process. The method described in this paper was only applied to FCOMs. However, it is usable for other procedures manuals (e.g. cookbooks or medical and surgery procedures). This could lead to the need for new roles.

This work can be continued in multiple ways. Concerning atomic instructions, a better framework to handle meta-instructions is needed. Working on the representation of physical contexts in which the actions are realised could be of great help. This can serve to better constrain the *hasEndLoc* property, for example.

Finally, to enhance this framework, extended studies of other corpuses, such as clinical procedures or cooking recipes, could help to generalize the ontology. The method presented is not dependent of the FCOM and could be applied on other documents.

Acknowledgments

This work was carried out as part of a second year master internship funded by Airbus - Service Engineering Computing & Communication Info & Data Processing.

References

[1] M. Grüninger, Using the PSL Ontology, in: S. Staab, R. Studer (Eds.), Handbook on Ontologies, Springer, 2009, pp. 423–443.
[2] Airbus A350 Flight Crew Operating Manual, 2019.
[3] L. C. Pouchard, A. Cutting-Decelle, J.-J. Michel, M. Grüninger, ISO 18629 PSL: A standardised language for specifying and exchanging process information, IFAC Proceedings Volumes 38 (2005) 37–45.
[4] M. Katsumi, M. Grüninger, Using PSL to extend and evaluate event ontologies, in: N. Bassiliades, G. Gottlob, F. Sadri, A. Paschke, D. Roman (Eds.), Rule Technologies: Foundations, Tools, and Applications, Springer, 2015, pp. 225–240.
[5] D. Lewis, Against structural universals, Australasian Journal of Philosophy 64 (1986) 25–46.
[6] D. M. Armstrong, In defence of structural universals, Australasian Journal of Philosophy 64 (1986) 85–88.
[7] J. Bigelow, Towards structural universals, Australasian Journal of Philosophy 64 (1986) 94–96.
[8] A. R. Fisher, Structural universals, Philosophy Compass 13 (2018) e12518.
[9] P. Garbacz, An analysis of the debate over structural universals, in: Formal Ontology in Information Systems, IOS Press, 2020, pp. 3–16.
[10] ISO 24617-4:2014, Language resource management – Semantic annotation framework (SemAF) – Part 4: Semantic roles (SemAF-SR), Standard, International Organization for Standardization, Geneva, CH, 2014.
[11] E. Jezek, L. Vieu, F. Zanzotto, G. Vetere, A. Oltramari, A. Gangemi, R. Varvara, Enriching 'senso comune' with semantic role sets, in: 10th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA 2014) in conjunction with LREC 2014, Association for Computational Linguistics (ACL), Reykjavik, IS, 2014, pp. 88–94.
[12] R. Arp, B. Smith, A. D. Spear, Building ontologies with Basic Formal Ontology, The MIT Press, 2015.
[13] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, The WonderWeb Library of Foundational Ontologies and the DOLCE ontology, WonderWeb (EU IST Project 2001-33052) Deliverable D18, LOA-ISTC-CNR, 2003.
[14] H. Herre, General Formal Ontology (GFO): A foundational ontology for conceptual modelling, in: Theory and applications of ontology: computer applications, Springer, 2010, pp. 297–345.
[15] G. Guizzardi, Ontological foundations for structural conceptual models, Ph.D. thesis, University of Twente, 2005.