Towards a Formal Argumentation-based Model for Procedural Texts

Leila Amgoud and **Farida Aouladomar** and **Patrick Saint-Dizier** [1]

Abstract.

In this paper, we first present an analysis of the facets of natural argumentation in procedural texts. Next, we extend the formal model proposed in [2] to accommodate these facets. Finally, we outline the main properties of the model.

1 Introduction

Procedural texts are specific forms of discourse, satisfying constraints of economy of means, accuracy, etc. They are in general based on a specific discursive logic, made up of presuppositions, causes and consequences, goals, inductions, warnings, anaphoric networks, etc., and more psychological elements (e.g. *to stimulate a user*). The goal is to optimize a logical sequencing of instructions and to make the user feel safe and confident with respect to the goal(s) he wants to achieve. This type of discourse contains a number of facets, which all are associated in a certain way to argumentation: procedural discourse is indeed informative, narrative, explicative, descriptive, injunctive and sometimes figurative. In fact, argumentation does provide a motivation and an internal coherence to procedural texts: procedural discourse is basically interactive: it communicates, teaches, justifies, explains, warns, forbids, stimulates, evaluates.

Procedural texts consist of a structure goal-subgoals or task-subtasks designed with some accuracy in order to reach an objective (e.g. assemble a computer). In our perspective, procedural texts range from apparently simple cooking receipes to large maintenance manuals (whose paper versions are measured in tons e.g. for aircraft maintenance). They also include documents as diverse as teaching texts, medical notices, social behavior recommendations, directions for use, do-it-yourself and assembly notices, itinerary guides, advice texts, savoir-faire guides, etc. [1].

More precisely, a procedural text is a structure composed of a main *goal* or task, which is decomposed recursively into subtasks. Leaves are elementary tasks, also called instructions. The *tree structure* reflects in general the temporal structure of the system in terms of temporal precedence. To make it more precise, we present below a model that allows for different temporal combinations of tasks (precedence, overlap, etc.). Finally, this tree structure also reveals the modularity of procedures, via the task-subtask decomposition. Therefore, constraints and arguments stated within a subtask, only range over the elements of that subtask.

The backbone of a procedural text is clearly the task-subtasks structure. In most types of procedural texts, procedural discourse has in fact two deeply intertwinned dimensions: an *explicative component*, constructed around rational and objective elements (the task-subtask structure), and a *seduction component* whose goal is (1) to motivate the user by outlining the importance of certain tasks and the necessity to fully realize them, by giving various forms of advices, (2) to make the user understand that the procedure proposed is a safe and efficient way to reach the goal, (3) to prevent the user from making errors of various types via warnings. This seduction component closely associated with the rational elements, forms, in particular, the argumentative structure of the procedural text.

The diversity of procedural texts, their objectives and the way they are written is the source of a large variety of natural arguments. This study is based on a extensive corpus study, within a language production perspective. This approach allows us to integrate logical, linguistic (e.g. [6, 3]) and philosophical views of argumentation.

The aim of this paper is to present a formal model for procedural texts. The model is an extension of a framework developed in [2] for reasoning about agent's desires. The idea is to built plans for achieving those desires, to resolve the conflicts among those plans, and finally to select the set of desires that are achievable together.

In the next sections of this paper, we present some details about a typology of arguments in procedural texts and a motivational example. Then, we present an extension of the formal model developed in [2] for modeling procedural texts and some essential properties.

2 Procedural texts and argumentation

2.1 Role of arguments

In [4], we present in detail the different linguistic and conceptual forms of arguments found in procedural texts. This is a study done for french. Let us review here the 5 major forms of arguments we found frequently in corpora. Verb classes referred to are in general those specified in WordNet:

- Explanations are the most usual ones. We find them in any kind of procedural texts. They usually introduce a set of sequences or more locally an instruction implemented in the "goal" symbol of the grammar. The abstract schemas are the following: (1) purpose connectors-infinitive verbs, (2) causal connectors-deverbals and (3) titles. The most frequently used causal connectors are : pour, afin de, car, c'est pourquoi, etc. (to, in order to) (e.g. to remove the bearings, for lubrification of the universal joint shafts, because it may be prematurely worn due to the failure of another component).
- Warning arguments embedded mostly either in a "negative" formulation. They are particularly rich in technical domains. Their role is basically to explain and to justify. Negative formulation is easy to identify: there are prototypical expressions that introduce the arguments. Negative formulation follows the abstract schemas:

[1] IRIT - CNRS 118, route de Narbonne 31062 Toulouse Cedex 9, France amgoud, aouladom, stdizier@irit.fr

negative causal connectors-infinitive risk verbs; negative causal marks-risk VP; positive causal connectors-VP negative syntaxic forms, positive causal connectors-prevention verbs.

  - negative connectors: sous peine de, sinon, car sinon, sans quoi, etc. (otherwise, under the risk of) (e.g. sous peine d'attaquer la teinte du bois).
  - risk class verbs: risquer, causer, nuire, commettre etc. (e.g. pour ne pas commettre d'erreur).
  - prevention verbs: viter, prvenir, etc. (e.g. afin d'viter que la carte se dchausse lorsqu'on la visse au chssis, gloss: in order to prevent the card from skipping off its rack).
  - Positive causal mark and negative syntaxic forms: de facon ne pas, pour ne pas, pour que ... ne ...pas etc. (in order not to) (e.g. pour ne pas le rendre brillant, gloss: in order not to make it too bright).

- Tip arguments: these arguments are less imperative than the other ones, they guide the user, motivate him, and help to evaluate the quality of the work. They are particularly present in communication texts. The corresponding abstract schemas are: causal connectors-performing NP; causal connectors-performing verbs; causal connectors-modal-performing verbs; performing proposition.
  - performing verbs: e.g. permettre, amliorer, etc. (allow, improve, etc.).
  - performing PPs: e.g. Pour une meilleure finition; pour des raisons de performances (for a better finishing, for performing reasons).
  - performing proposition: e.g. Have small bills. It's easier to tip and to pay your bill that way.
- threatening arguments and reward arguments: these arguments have a strong impact on the user's intention to realize the instruction provided, the instruction is almost made compulsory by using this kind of argument. This is the injunctive form. We could not find any of these types of arguments in procedural texts, except in QA pairs and injunctions texts (e.g. rules) where the author and the adressee are clearly identified. Therefore, in those arguments we often find personal pronouns like "nous" "vous" (we, you). For threatening arguments, it follows the following schemas: otherwise connectors-consequence proposition; otherwise negative expression-consequence proposition :
  - otherwise connectors: sinon.
  - otherwise negative expression: si ... ne ... pas... (e.g. si vous ne le faites pas, nous le primerons automatiquement aprs trois semaines en ligne).
- For reward arguments, the schemas associated are the following : personal pronouns - reward proposition :
  - reward proposition : using possession transfer verbs (gagner, donner, bnficier, etc. (win, give, benefit)

Besides these five main types of arguments, we found some forms of stimulation-evaluation (what you only have to do now...), and evaluation.

2.2 The pragmatic dimensions of argumentative aims

First, it is important to note that arguments associated to a task do not form a homogeneous group. Arguments have different types (as specified above), and range over various facets of the task to carry out. We can talk, similarly to the temporal organization where some tasks may evolve in parallel with little connections, of a polyphony of arguments, to be contrasted with a sequence of arguments jointly operating over the same set of data.

Another important aspect is that, besides their direct use and meaning, arguments or groups of arguments convey several pragmatic effects which are quite subjective. For example, a task may become more salient in the procedural discourse if it is associated with a large number of arguments. Arguments therefore may induce zoom effects on some instructions. Arguments are also often exaggerated, beyond normal expectations, as a way to strengthen them, and to arouse a greater attention from the reader. In texts dedicated to the large public, arguments may be too strong, in particular precautions to take (a form of warning). The result is that the global coherence of arguments over a whole procedural text may not be fully met, while the text remains perfectly 'coherent' from the reader point of view. Finally, a task may be associated with a disjunction of arguments, whose selection depends on the reader's performances and preferences, for example tips and explanations may be tuned to various audiences.

These short considerations are illustrated below. They obviously have an impact on the formal model, in which we will need to introduce temporal dimensions, flexible forms of coherence, locally and globally, and over the various types of arguments (e.g. a tip must not contradict a warning), preferences and salience effects.

3 Illustrative example (Assembling a PC)

Let us illustrate the previous section by means of a simple, real example, extracted from the Web, which will be used throughout the remainder of this paper. The example is about *assembling a PC*. The following instructions are given for that purpose:

Assembling your PC

Material required: Make sure that you have all the below materials before starting: Processors, Motherboard, Hardisk, RAM, Cabinet, Floppy drive, . . .

Precautions: Before starting the actual assembly, the following precautions would help to avoid any mishap during the assembly process:

- be sure to handle all the components with great care, . . .
- use a clean and large enough table, . . .
- avoid the presence of any source of static electricity around,. . .

Procedure:

- *Installing Hardisk*: Ensure that the hard drive is set up to be the master drive on its IDE cable. If so plug it in . . .
- *Floppy Drive*: Plug in the power cable (see picture) carefully since it is quite possible to miss one of the connectors, which will quite possibly cause some damage when the computer is powered on. Then, place drive in slot, . . .

As the reader can note it, procedural texts contain a large number of arguments under the form of advices, warnings, etc. which do help to realize the action. Note also the elliptical style of some titles, with no verbs, but which are nevertheless actions.

4 Logical language

Let $\mathcal{L}$ be a logical language, and $\texttt{Arg}(\mathcal{L})$ the different arguments that can be built from $\mathcal{L}$. From $\mathcal{L}$, three bases can be distinguished:

- $\mathcal{G}$ contains formulas of $\mathcal{L}$. Elements of $\mathcal{G}$ represent the *subject* or the *goals* to be satisfied through the procedural text. For instance, the goal of the procedural text given in Example 1 is "assembling a PC". Note that, for the same text, the set $\mathcal{G}$ should be consistent.
- $\mathcal{P}$ contains rules having the form $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \varphi$ where $\varphi_1, \ldots, \varphi_n, \varphi$ are elements of $\mathcal{L}$. Such a formula means that the author believes that if the actions $\varphi_1, \ldots, \varphi_n$ are achieved then φ will also be achieved.

Example 1 *In the above example, the different bases contain among others the following information:* $\mathcal{G}$ = {*Assembling the Computer*}. $\mathcal{P}$ = { *Check Required Material* $\wedge$ *Installing Hard Drive* $\wedge$ *Floppy Drive* $\rightarrow$ *Assembling the Computer, Processors* $\wedge$ *Motherboard* $\wedge$ *Hardisk* $\wedge$ *RAM* $\wedge$ *Cabinet* $\wedge$ *Floppy Drive* $\rightarrow$ *Check Required Material set up the harddisk* $\wedge$ *plug in* $\rightarrow$ *Installing HardDisk, plug in the power cable* $\wedge$ *place drive in slot* $\rightarrow$ *Floppy Drive*}.

5 A basic argumentation system

A rational agent can express claims and judgments, aiming at reaching a decision, a conclusion, or informing, convincing, negotiating with other agents. Pertinent information may be insufficient or conversely there may be too much, but partially incoherent information. In case of multi-agent interactions, conflicts of interest are unavoidable. Agents can be assisted by argumentation, a process based on the exchange and the valuation of interacting arguments which support opinions, claims, proposals, decisions, etc. According to Dung [5], an argumentation framework is defined as a pair consisting of a set of arguments and a binary relation representing the defeasibility relationship between arguments.

Definition 1 *(Argumentation framework) An* argumentation framework *is a pair* $<\mathcal{A}, \mathcal{R}>$ *where* $\mathcal{A}$ *is a set of arguments (*$\mathcal{A} \subseteq \texttt{Arg}(\mathcal{L})$*), and* $\mathcal{R}$ *is a binary relation representing a defeasibility relationship between arguments, i.e.* $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$*.* $(a, b) \in \mathcal{R}$ *or equivalently* $a\mathcal{R}b$ *means that the argument* a *defeats* b*.*

In the above definition, the structure of the argument is unknown. In the remainder of this paper, we do not need to define formally an argument. However, any argument $a \in \mathcal{L}$ is supposed to have a conclusion that is returned by the function $\texttt{Conc}$.

Since arguments may be conflicting, it is important to know which arguments are considered *acceptable*. Dung has defined different acceptability semantics.

Definition 2 (Defence/conflict-free) *Let* $S \subseteq \mathcal{A}$*.*

- S defends *an argument* A *iff each argument that defeats* A *is defeated by some argument in* S*.*
- S *is* conflict-free *iff there exist no* A_i*,* A_j *in* S *such that* A_i *defeats* A_j*.*

Definition 3 (Acceptability semantics) *Let* S *be a conflict-free set of arguments and let* $\mathcal{F}$*:* $2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ *be a function such that* $\mathcal{F}(S) = \{A \mid S \text{ defends } A\}$*.*

- S *is a* complete *extension iff* $S = \mathcal{F}(S)$*.*
- S *is a* preferred *extension iff* S *is a maximal (w.r.t set* $\subseteq$*) complete extension.*
- S *is a* grounded *extension iff it is the smallest (w.r.t set* $\subseteq$*) complete extension.*

Note that there is only one grounded extension. It contains all the arguments that are not defeated, and the arguments that are defended directly or indirectly by non-defeated arguments.

The last step of an argumentation process consists of determining, among all the conclusions of the different arguments, the "good" ones called *justified conclusions*. Let $\texttt{Output}$ denote this set of justified conclusions. One way of defining $\texttt{Output}$ is to consider the conclusions that are supported by at least one argument in each extension.

Definition 4 (Justified conclusions) *Let* $(\mathcal{A}, \mathcal{R})$ *be an argumentation system and* $\{E_1, \ldots, E_n\}$ *be its set of extensions (under a given semantics).* $\texttt{Output} = \{\psi | \forall E_i, \exists A \in E_i$ *such that* $\texttt{Conc}(A) = \psi\}$*.*

6 A formal model for procedural texts

In this section we propose a formal framework for procedural texts. This framework builds on a model developed in [2] for reasoning about conflicting desires. The basic idea behind that model is to construct plans for each desire and then to select the set of desires that are achievable together. A plan consists in decomposing the initial desire into sub-desires that are themselves decomposed into other sub-desires. This gives birth to a tree structure, where the leaves of the tree are instructions. In what follows, we will adopt this notion of plan for modeling a procedural text.

The basic concept of our framework is that of *goal*. Indeed, each procedural text is supposed to have a goal. A goal is any element of $\mathcal{G}$, and it may have sub-goals.

Definition 5 (Goal/Sub-goal) *Let us consider the bases* $<\mathcal{G}, \mathcal{P}>$*.*

1. $\mathcal{G}$ *is the set of* goals*.*
2. $Sub\mathcal{G}$ *is the set of the* sub-goals*: A literal* $h' \in Sub\mathcal{G}$ *iff there exists a rule* $\varphi_1 \wedge h' \ldots \wedge \varphi_n \rightarrow \varphi \in \mathcal{P}$ *with* $\varphi \in \mathcal{G}$ *or* $\varphi \in Sub\mathcal{G}$*. In that case,* h' *is a sub-desire of* φ*.*

A goal can be achieved in different ways. We bring the two notions together in a new notion of *partial plan*.

Definition 6 (Partial plan) *A* partial plan *is a pair* $a = <h, H>$ *such that:*

- h *is a goal or a sub-goal.*
- $H = \{\varphi_1, \ldots, \varphi_n\}$ *if there exists a rule* $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow h \in \mathcal{P}$*,* $H = \emptyset$ *otherwise.*

The function $\texttt{Goal}(a) = h$ *returns the goal or sub-goal of a partial plan* a*.* $\aleph$ *will gather all the partial plans that can be built from* $<\mathcal{G}, \mathcal{P}>$*.*

Remark 1 *A goal may have several partial plans corresponding to different alternatives for achieving that goal. Indeed, in procedural texts, it may be the case that for the same goal/sub-goals, several ways for achieving it are provided.*

Remark 2 *Let* $a = < h, H >$ *be a partial plan. Each element of the support* H *is a sub-goal of* h*.*

Definition 7 *A partial plan $a = <h, H>$ is* elementary *iff $H = \emptyset$.*

Remark 3 *If there exists an elementary partial plan for a goal h, then this means that the agent knows how to achieve h directly. This corresponds to the notion of instructions of procedural texts.*

Example 2 *In the above example, the following partial plans can be built: <Assembling the Computer, {Check Required Material, Installing Hard Disk, Floppy Drive}>, <Check Required Material, {Processors, Motherboard, Hardisk, RAM, Cabinet, Floppy Drive}>, <Installing Hard Disk, {Set up the harddisk, plug in}>, <Floppy Drive, {plug in the power cable, place drive in slot}>.*

A partial plan shows the actions that should be performed in order to achieve the corresponding goal (or sub-goal). However, the elements of the support of a given partial plan are considered as sub-goals that must be achieved in turn by another partial plan. The whole way to achieve a given goal is called a *complete plan.* A *complete plan* for a goal d is an *AND* tree. Its nodes are partial plans and its arcs represent the sub-goal relationship. The root of the tree is a partial plan for the goal d. It is an AND tree because all the sub-goals of d must be considered. When for the same goal, there are several partial plans to carry it out, only one is considered in a tree. Formally:

Definition 8 (Complete plan) *A* complete plan *G for a goal h is a finite tree such that:*

- *$h \in \mathcal{G}$.*
- *The root of the tree is a partial plan <h, H>.*
- *A node <h', $\{\varphi_1, \ldots, \varphi_n\}$> has exactly n children $<\varphi_1, H'_1>$, $\ldots$, $<\varphi_n, H'_n>$ where $<\varphi_i, H'_i>$ is an element of $\aleph$.*
- *The leaves of the tree are elementary partial plans.*

The function $\texttt{Nodes}(G)$ returns the set of all the partial plans of the tree G. $\mathcal{CP}$ denotes the set of all the complete plans that can be built from $<\mathcal{G}, \mathcal{P}>$. The function $\texttt{Leaves}(G)$ returns the set of the leaves of the tree G.

Example 3 *In our training example, there is a unique complete plan for the goal "assembling a PC" that is shown in Figure 1.*

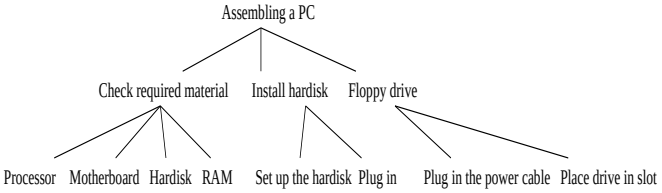

Figure 1. Complete plan

Note that a procedural text may have several complete plans capturing the different ways for achieving the goal of the text.

As said in the introduction, a procedural text may contain arguments for explaining different tasks, and for motivating the reader to behave in a certain way. In [4], we have shown there are mainly tow categories of arguments that are used in procedural texts: *advices* and *warnings*. It is also common that some arguments in a procedural text may defeat other arguments. Indeed, some authors explain a task, and present for that purpose arguments. Since, the authors may expect counter-arguments from the readers, then they introduce those counter-arguments in the text itself and present the counter-attack against them. In sum, the tasks of the procedural text should be justified, and defended in the text.

Now that all the ingredients introduced, we are ready to define formally a procedural text. Indeed, a procedural text has three components: a *goal* that it should satisfy, a *complete plan* for achieving that goal, and an argumentation system that justifies each goal/sub-goal occurring in the complete plan.

Definition 9 (Procedural text) *A* procedural text *is a tuple <g, G, AS> where:*

- *$g \in \mathcal{G}$ is the goal of the procedural text*
- *$G \in \mathcal{CP}$ is a complete plan for g*
- *$AS = <\mathcal{A}, \mathcal{R}>$ is an argumentation system*
- *$\forall\ a_i \in \texttt{Node}(G), \texttt{Goal}(a_i) \in \texttt{Outcome}(AS)$*

The last condition ensures that the procedural text is coherent in the sense that each goal and sub-goal is justified and correctly supported. This means that the arguments exchanged for supporting a given goal/sub-goal cannot defeat another goal/sub-goal.

7 Conclusion

This paper has proposed a formal model for defining procedural texts. We have mainly shown how these texts can be defined in a more abstract way. Due to the tree structure of procedural texts and their decomposition in terms of tasks and sub-tasks, we have defined a procedural text as a plan for achieving its goal. This formal model makes it possible to easily compare different procedural texts. For instance, a procedural text in which the set $\mathcal{A}$ of arguments is empty is poor, and may be directed towards a professional audience. Therefore further investigations should be carried out in order to study different strategies an author may use according to target audiences.

REFERENCES

[1] J. M. Adam, 'Types de textes ou genres de discours ? comment classer les textes qui disent de et comment faire', *Langages*, **141**, 10–27, (2001).

[2] L. Amgoud, 'A formal framework for handling conflicting desires', in *Proceedings of the 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'2003*, pp. 552–563, (2003).

[3] J.-Cl. Anscombre and O. Ducrot, 'Interrogation et argumentation', *Langue franaise*, **52, L'interrogation**, (1981).

[4] F. Aouladomar, L. Amgoud, and P. Saint-Dizier, 'On argumentation in procedural texts', in *Proceedings of the International Symposium: Discourse and Document*, (2006).

[5] P. M. Dung, 'On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games', *Artificial Intelligence*, **77**, 321–357, (1995).

[6] J. Moeschler, 'Argumentation et conversation, éléments pour une analyse pragmatique du discours', *Hatier - Credif*, (1985).