Applied Intelligence (2023) 53:26351–26363
https://doi.org/10.1007/s10489-023-04882-0

Mapping natural language procedures descriptions to linear temporal logic templates: an application in the surgical robotic domain

Marco Bombieri[1] · **Daniele Meli**[1] · **Diego Dall'Alba**[1] · **Marco Rospocher**[1] · **Paolo Fiorini**[1]

Accepted: 10 July 2023 / Published online: 22 August 2023

Abstract
Natural language annotations and manuals can provide useful procedural information and relations for the highly specialized scenario of autonomous robotic task planning. In this paper, we propose and publicly release AUTOMATE, a pipeline for automatic task knowledge extraction from expert-written domain texts. AUTOMATE integrates semantic sentence classification, semantic role labeling, and identification of procedural connectors, in order to extract templates of Linear Temporal Logic (LTL) relations that can be directly implemented in any sufficiently expressive logic programming formalism for autonomous reasoning, assuming some low-level commonsense and domain-independent knowledge is available. This is the first work that bridges natural language descriptions of complex LTL relations and the automation of full robotic tasks. Unlike most recent similar works that assume strict language constraints in substantially simplified domains, we test our pipeline on texts that reflect the expressiveness of natural language used in available textbooks and manuals. In fact, we test AUTOMATE in the surgical robotic scenario, defining realistic language constraints based on a publicly available dataset. In the context of two benchmark training tasks with texts constrained as above, we show that automatically extracted LTL templates, after translation to a suitable logic programming paradigm, achieve comparable planning success in reduced time, with respect to logic programs written by expert programmers.

Keywords Natural language processing · Autonomous planning · Linear temporal logic · Surgical robotics

1 Introduction

Robots are becoming increasingly used in complex domains involving interaction with humans, such as surgery, manufacturing, and education. In these domains, safe and trustable autonomy is a key objective [1]. This can be achieved by adopting a formal representation of task knowledge, e.g., with the Planning Domain Description Language (PDDL) [2] and logic programming [3] implementations, defining task resources and specifications (preconditions and effects of actions). However, in complex domains, task knowledge is not easily available to robotic programmers. Recent research articles [4, 5] have shown that Natural Language Processing (NLP) for automatic extraction of task knowledge from texts is a promising approach to retrieving relevant procedural information about a given domain, thus mitigating the effort for robot programmers. However, existing NLP techniques often make simplifying assumptions on the semantic and syntactic richness of the input texts and the domain itself, thus struggling in realistic robotic scenarios where complex temporal and logical relations are involved to describe the flow of actions and events.

In this paper, we propose AUTOMATE (l**A**ng**U**age **T**o l**O**gic te**M**pl**ATE**s), a pipeline for the automatic extraction of procedural Linear Temporal Logic (LTL) [6] templates from texts. Our methodology combines automatic detection of procedural sentences, Part-Of-Speech (POS) tagging combined with Semantic Role Labeling (SRL) to extract relevant semantic information about task knowledge, and recognition of LTL connectors according to domain-dependent language constraints. We present and validate our methodology in the complex yet unexplored scenario of surgical robotics, which involves highly specialized procedural descriptions. We select this domain because, in the surgical context, autonomy

These authors contributed equally to this work.

✉ Marco Bombieri
marco.bombieri_01@univr.it

✉ Daniele Meli
daniele.meli@univr.it

[1] University of Verona, Verona, Italy

has the potential to reduce patients' recovery time, usage and cost of hospital resources, and surgeon's fatigue [7]. This paper specifically investigates the following research questions:

[RQ1] Is the automatic translation of procedural textual descriptions to LTL templates a feasible task without relying on too strict and unrealistic language constraints?
[RQ2] If the answer to RQ1 is positive, can the obtained templates be automatically translated to logic programs for direct implementation in an autonomous robotic architecture?
[RQ3] Is there any advantage, e.g., in terms of performance, in using task knowledge extracted from texts rather than expert-written logic programs?

We then make the following contributions to the state of the art:

- in the exemplary surgical context, we show how to perform a systematic linguistic and stylistic analysis of domain-specialized texts, to define realistic language constraints (about, e.g., verbal forms and tenses, LTL connectors, and semantic procedural structures) and preserve domain expressiveness;
- we propose AUTOMATE for automatic generation of LTL templates directly from realistic natural language descriptions, and make it publicly available at https://gitlab.com/altairLab/AUTOMATE;
- we show how to combine LTL templates with commonsense and domain-independent knowledge to implement a robotic-executable logic program and bridge the gap between NLP and real robotic applications;
- in the context of two benchmark tasks for surgical robotics, namely, peg transfer and tissue retraction, we write texts following the constraints defined before, and we show that extracted LTL templates are as successful as expert-written logic programs, with the former often being more computationally efficient.

The paper is organized as follows: Section 2 revises the state of the art in procedural knowledge extraction from text, highlighting recent solutions for extracting LTL relations. Section 3 shows how to define domain-specific language constraints, particularly for surgical robotics. Based on them, texts for our benchmark scenarios are presented in Section 4, and Section 5.2 details AUTOMATE. Section 6 evaluates the planning performance of extracted LTL templates against expert-written logic programs. Finally, Section 6.1.1 summarizes the responses to our research questions, highlighting the benefits and limitations of AUTOMATE and possible future research directions.

2 Related works

The focus of this paper is on machine understanding of procedural task descriptions expressed in natural language. As analyzed in [8], this problem is becoming relevant in many different domains and for many purposes, such as question-answering, intent inference (i.e., prediction of the goal of a sequence of actions using commonsense reasoning), task-based search, activity recognition, and language grounding (e.g., alignment of natural language text to videos or images). Recent research has also explored retrieval of LTL relations from texts, as reviewed in [9]. In this section, we analyze the most recent works focusing on this latter aspect, comparing AUTOMATE to them in terms of required input, output, the domain of application, and limitations.

Machine understanding of procedural task descriptions is relevant to domains involving repair instructions [10–12], technical support documentation [11], cooking recipes [11, 13], construction procedures [14, 15], and business process modeling [16]. Only a few recent papers have investigated the problem of mapping procedural knowledge extracted from text to formal logic [5, 17].

In [11], sentences mentioning actions in cooking recipes and maintenance manuals are detected with a convolutional neural network fed with word embeddings. The goal of the authors is to build a procedural workflow, thus recognizing actions, generic roles and objects, parts of the text where a procedural block begins or ends, and clues about an action that is optional or can be executed concurrently with another one. This method thus cannot extract LTL templates (the output is a list of sentences or classified tokens), nor recognize the precise semantics of an identified object (for example the mention of an instrument). [12] deals with procedural sentence understanding in repair instructions, intending to extract verbs, tools, and disassembled parts by using BERT-based methods. Its purpose is however that of recognizing procedural actors in texts and not that of extracting LTL relationships or an executable workflow. Finally, also [13] deals with procedural understanding in cooking recipes by releasing an annotated dataset; however, its main goal is to ground dialogues in which agents, given a recipe document, guide the user to cook a dish, to provide a framework for understanding users' questions and generating automatic responses. In [10], the authors investigate different combinations of features (e.g., bags of words, post length, bullet lists) and machine learning methods (e.g., neural networks and random forests) to detect posts from automotive web communities containing descriptions of repair instructions. However, the final goal is not to extract LTL specifications or procedural workflows. In [14], the authors, with a named entity recognition system combined with a relations extraction method, target construction regulatory Chinese documents, to extract procedural temporal constraints between events, e.g., identifying if two of

Table 1 Summary of related works, compared to ours. AUTOMATE is the only methodology that automatically extracts LTL relations from specialized natural text

Paper	Domain	Input	Output
[11]	Cooking instructions and maintenance manuals	Elementary sentences (subject-verb-object)	Sequence of actions
[12]	Repair manuals and cooking instructions	Elementary sentences (subject-verb-object)	List of actions, tools and disassembled objects
[13]	Cooking instructions	Dialogues about procedures	Q&A system based on language grounding
[10]	Repairing instructions	Web posts	Classification of procedural content
[14]	Chinese construction regulations	Chinese natural language	Flow chart
[15]	Construction safety regulations	Natural language	Query graphs
[16]	Business process modeling	E-mails and chats	Sequence of actions
[17]	General purpose	Natural language	Sequence of events
[5]	Robotic task instructions	Structured semantic tuples	LTL relations
Ours	(Surgical) robotic procedures	Procedural natural language	LTL relations

them have to be conducted simultaneously or which of them starts before. Similarly, [15] extracts safety requirements from construction regulatory documents using a CNN-based model and represents them in the form of knowledge graph-based queries. Anyway, the input language and final goal of both the previous papers are different from ours. The Business Process Modeling and Notation (BPMN) community is also investigating the possibility of extracting structured workflows from unstructured procedural documents in natural languages, such as e-mails and chats: the model proposed by [16], e.g., takes as input a document (an email or chat containing instructions), identifies and clusters main actions, and constructs structured and time-ordered business event logs. However, still, a workflow is only extracted.

In the range of papers dealing with LTL rules extraction from text, we mention [17], whose goal is to order events extracted from textual descriptions. While relevant, their pipeline does not target automation of procedures, but it simply constructs temporal sequences of events, rather than identifying actions, LTL constructs, and relations. [5] also deals with the mapping of task description to LTL rules; however, the input is a rigidly structured signature, rather than natural language.

All the mentioned works differ from ours, both for the purpose and thus expected output. Moreover, the language constraints they considered are often very strict and simplifying. For these reasons, a direct comparison is not possible. Nevertheless, Table 1 qualitatively compares these related works, summarizing relevant information.

3 Definition of domain language constraints

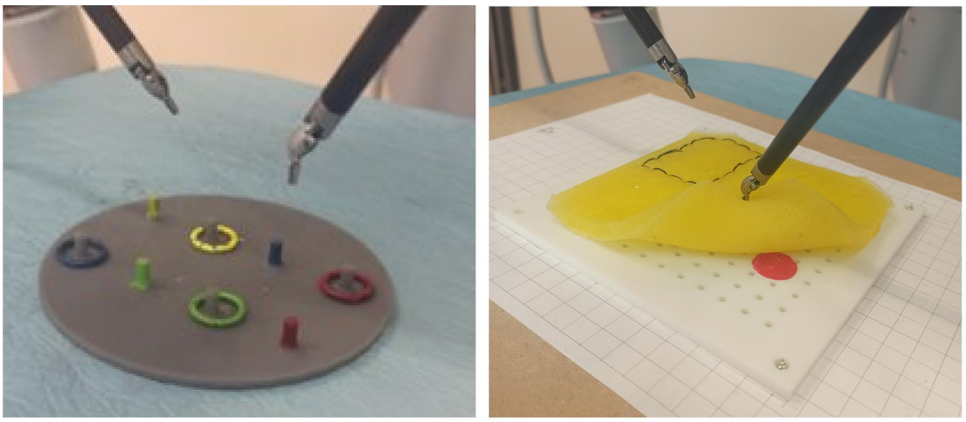
The input to AUTOMATE pipeline is a domain-specific text adhering to some lightweight language constraints. Specifically, these constraints define how the following concepts are expressed:

- *description of robot setup*, e.g., instruments' docking; this information is relevant to identify *agents of the task*;
- *action representation*, i.e., how operations of the procedure are expressed in domain language, including predicates (verbs) with relevant semantic roles;
- *causal and temporal flow* of the task, i.e., necessary *conditions* and *temporal sequences* of actions, as well as *loops* defining continuation of (sequences of) actions.

To preserve the rich expressiveness of highly specialized texts and address **RQ1**, it is important to define them with a domain-specific linguistic analysis, which can be easily replicated on texts from different domains. In this paper, we then consider the publicly available SPKS dataset [18], containing as-is sentences from different surgical task descriptions, taken from available textbooks and manuals. We analyze how the above concepts are described in SPKS, and set the most frequent patterns as language constraints for input texts to AUTOMATE, avoiding all the infrequent expressions.

Our analysis leads to the definition of the following domain-specific language constraints:

- the *robotic setup* is described in the first paragraph; specifically, robotic arms are presented with incremental numbering (e.g. *first arm* and *second arm*), while docking of instruments to the robotic arms is introduced by verbs as *equip*, *mount* and synonyms;
- *actions* (verbs) are expressed in active or passive form, at present or imperative tense. Verbs such as *use* and synonyms are allowed to introduce the main action; moreover, instruments may or may not coincide with *agents*, i.e., subjects of the actions, since sometimes the surgeon is subject;
- *conditions* can be only expressed with *if/otherwise* and *in case/otherwise* statements; *temporal sequences* can only contain *then* and *once* connectors; *loop iterations* can



(a) Peg transfer. (b) Tissue retraction.

Fig. 1 The setup for the benchmark surgical training tasks with da Vinci. In the right figure, ROI is in red, and APs are sewed in the top left

only be expressed with *until-repeat* constructs. Finally, standard logical connectors, such as *and, or*, are commonly used to specify alternative workflows.

As a final remark, the use of *synonyms* in our texts is limited to those statically recognized by the state-of-the-art WordNet resources [19]. For the scope of this paper and the considered benchmark tasks, this is a reasonable assumption. When more complex and domain-specific terminology is needed, e.g., in very complex surgical tasks or other more specialized domains, a refinement of WordNet is possible, as proposed in [20].

4 Benchmark tasks and texts

Before detailing AUTOMATE, we introduce texts for our experimental evaluation, related to the benchmark tasks of *peg transfer* and *tissue retraction*, following the language constraints defined above. This will be useful to illustrate the main aspects of our pipeline with examples.

The setup for both tasks consists of three patient-side *arms* of the research version of the da Vinci surgical robot, namely the da Vinci Research Kit (dVRK) [21]. Two arms are equipped with graspers (first arm and second arm) and one holds the camera. The peg transfer (Fig. 1a) is a training task from the Fundamentals of Laparoscopic Surgery (FLS) [22], recognized as a benchmark for performance assessment in autonomous robotic surgery [23]. Tissue retraction (Fig. 1b) is a benchmark task for evaluating the performance of autonomous surgical systems [24, 25].

For both tasks, we consider two different kinds of procedural texts, according to the *agent's perspective*:

- *robot as agent*, where the instruments or the robotic arms are subjects of the sentences.
- *surgeon as agent*, where the surgeon teleoperates the robot, hence he is the subject.

While the second perspective is the dominant one in the SPKS dataset, testing also the *robot-as-agent* case allows us to validate the robustness of AUTOMATE to changes in semantic roles.

4.1 Peg transfer

PEG TRANSFER - Robot as agent

The setup has three arms. The first and second arms are equipped with grippers, while the third arm has a camera mounted on it for vision. 4 rings of different colors (red, green, blue, yellow) are placed on a base with 4 colored pegs and 4 grey pegs. First, the camera identifies the rings. Then, the first and second arms open the grippers. The camera selects one colored ring in the scene. If the ring is close to the first arm, the first arm attains it; otherwise, the second arm reaches the ring. Then, the gripper grasps the ring. Once the ring is on a peg, the arm raises it. Then, if the peg with the same ring's color is close to the arm, the arm reaches it; otherwise, it transfers the ring to the other arm. If the gripper is at the peg, the ring is placed on the peg. Then, the arm opens the gripper and goes to home position. The camera selects a ring to grasp and the procedure repeats until all visible rings are not on the same-colored pegs.

PEG TRANSFER - Surgeon as agent

The setup has three arms. The first and second arms are equipped with grippers, while the third arm has a camera mounted on it for vision. 4 rings of different colors (red, green, blue, yellow) are placed on a base with 4 colored pegs and 4 grey pegs. First, the surgeon identifies rings via camera. Once rings are detected, the surgeon opens the grippers of first and second arms and uses camera to select one colored ring in the scene. If the ring is close to the first arm, the surgeon uses the first arm to reach it; otherwise, the second arm is used to reach the ring. Once reached, the surgeon employs the grippers to grasp the ring. If the ring is on a peg, with help of the arm the surgeon raises it. Then, if the peg with the same ring color is close to the arm, they use grippers to reach it; otherwise, the ring is transferred to the other arm. If the gripper is at the peg, the surgeon places the ring on the peg, then opens the gripper and moves the arms to home position. The surgeon finally uses third arm to identify a ring to grasp and the procedure repeats until all rings are not on the same-colored pegs.

4.2 Tissue retraction

TISSUE RETRACTION - Robot as agent

The setup consists of three robotic arms. First arm and second arms are equipped with grippers, while third arm holds a camera for vision. A flap of adipose tissue is attached to surrounding anatomies at some points (APs), and covers a region of interest (ROI). The camera identifies the APs. First and second arms open the grippers. The camera selects a point on the tissue if it is far from APs. In case the point is close to first arm, the point is reached by first arm; otherwise, the second arm reaches the point. Then, the gripper grasps the tissue and raises it up. The arm lifts the tissue until a maximum height is reached, or maximum force is reached, or the ROI is visible. If the ROI is not visible in case of raising, the gripper goes towards the centre of tissue, horizontally. If the ROI is still not visible, the arm opens the gripper and goes upwards, the third arm selects a different grasping point and the procedure is repeated.

TISSUE RETRACTION - Surgeon as agent

The setup consists of three robotic arms. First arm and second arms are equipped with grippers, while third arm holds a camera for vision. A flap of adipose tissue is attached to surrounding anatomies at some points (APs), and covers a region of interest (ROI). The surgeon uses camera to identify APs. Then, the surgeon opens the first and second arm grippers. The surgeon exploits camera in order to select a point on the tissue if it is far from APs. If the point is close to first arm, the first arm is used to reach it; otherwise, the surgeon uses the second arm to reach the point. Then, the surgeon grasps the tissue with thr gripper and raises it. Using the arm, the surgeon lifts the tissue until a maximum height is reached, or maximum force is reached, or the ROI is visible. If the ROI is not visible in case of raising, the surgeon moves the gripper towards the centre of tissue horizontally. If the ROI is still not visible, the surgeon opens the gripper and moves it upwards, the surgeon uses the camera arm to select a different grasping point and the procedure is repeated.

5 AUTOMATE pipeline

This section describes our automatic pipeline to extract LTL templates from the benchmark text descriptions. A schematic representation is shown in Fig. 2. AUTOMATE consists of three main steps:

- automatic identification of procedural-only sentences and the description of the robotic setup, with the definition of the agents of the task (Section 5.1);
- SRL combined with POS tagging and semantic rules to automatically detect actions, agents and main semantic roles, and temporal-causal flows (Section 5.2);
- automatic translation of procedural semantic information to LTL rules (Section 5.3).

Since LTL templates are abstract representations of task knowledge, in Section 5.4 we show how to combine them with available commonsense and often domain-independent knowledge, in order to implement an effective logic program for robotic task planning.

5.1 Identifying robotic setup and procedural sentences

AUTOMATE first extracts robotic setup information. As explained in Section 3, this knowledge is contained in the first paragraph of surgical text descriptions. In general, this is not a strict requirement, since surgical instruments are common to most procedures and they can be retrieved from (task-independent) domain ontologies, e.g., [26][1]. AUTOMATE then searches for *equip, mount* and synonym verbs listed in Section 3, and exploits PropBank-based SRL [28] to match these verbs to semantic roles mentioning arms and instruments, in order to establish a static link between them (see Section 5.2 for more details on SRL). As an example, consider the text for the peg transfer task, with the surgeon as agent:

> The setup has three arms.
> The first and second arms are equipped with grippers, while the third arm has a camera mounted on it for vision.

Here, SRL recognizes verbs *equipped* and *mounted*. Moreover, it labels their subjects (the arms) and the closest semantic roles (*with grippers* and *a camera*. Then, after recognizing *grippers*, *camera* and the arms as elements in the list of instruments, AUTOMATE establishes a static link between *the first and second arms* and *grippers*, and between *thirs arm* and the *camera*. As a consequence, in the procedural description, they will be used interchangeably and recognized as synonyms.

Afterward, AUTOMATE incorporates the methodology proposed in [18] to remove non-procedural sentences from texts. In particular, a BERT-based classifier [29] pre-trained on the SPKS dataset is used to select only procedural sentences. For instance, consider the following sentence from the text for tissue retraction with the robot as an agent:

A flap of adipose tissue is attached to surrounding anatomies at some points (APs) and covers a region of interest (ROI).

This sentence just describes the anatomical setting and is classified as non-procedural, thus it is not processed further.

5.2 Procedural knowledge extraction

For each identified procedural sentence, AUTOMATE extracts actions and relevant semantic information, representing the procedural knowledge needed for LTL template

[1] Similar ontologies exist also for other robotics and automation domains, e.g., industry [27].

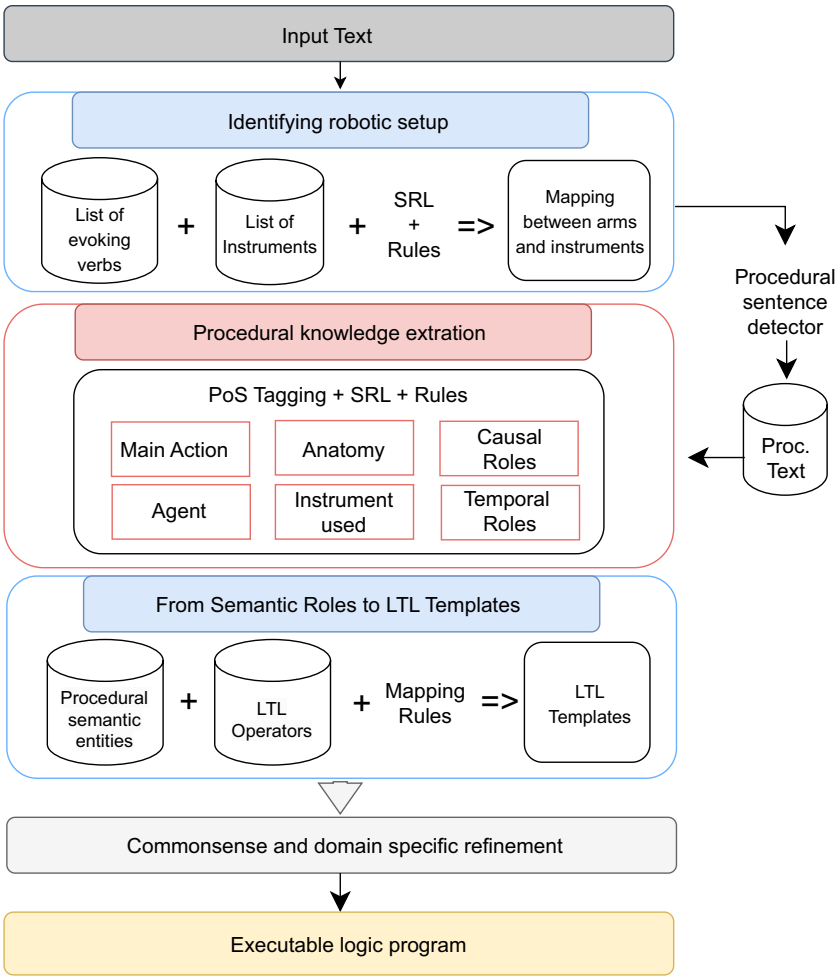


Fig. 2 Overview of the proposed approach: AUTOMATE to extract LTL templates + semi-automatic translation to executable logic programs

extraction. To this purpose, it still exploits the same PropBank-based SRL method [28] combined with a standard POS-tagger [30], and a specific filtering system for LTL connectors based on the language constraints described in Section 3. Details are reported in the next sections.

5.2.1 Selecting main action

While we allow multiple verbs to occur in a sentence, only one actually can describe an action of the procedure. Given a procedural sentence, AUTOMATE first uses a state-of-the-art POS tagger [30] to identify verbs, hence potential actions. Then, from language constraints, it only keeps verbs tagged with active, passive, present, or imperative tenses tags, thus excluding *-ing* forms, modals, and auxiliaries. Finally, it also excludes *use* verb and its synonyms, because they can only be used to introduce the main action according to SPKS-based constraints. Though SRL can identify verbs as well, the POS tagger also returns the tense, thus making verb filtering straightforward. AUTOMATE then excludes all the remaining candidate verbs that appear in a span of text labeled as causal or temporal role by SRL (see next section) through a list of static rules.

The following excerpt from the text for peg transfer with the surgeon as agent exemplifies the process:

If the ring is close to first arm, the surgeon uses first arm to reach it; [...]

Three verbs are identified in this sentence, i.e., *is*, *uses*, and *reach*. However, only *reach* is identified as the main action by AUTOMATE, while *uses* is filtered out and *is* gets discarded because it appears in a causal condition (introduced by *if*).

5.2.2 Identifying semantic roles

In order to recognize the *agent* performing the action, the *object* (e.g., the anatomy) undergoing it and the *causal and temporal* (LTL) task relations, AUTOMATE uses the state-of-the-art PropBank-based SRL method [28]. This method labels relevant semantic roles in a sentence, using tags as *Arg0* for the subject, *Arg1* for the object, *ArgM-ADV* for adverbial modifiers, *ArgM-DIS* for discourse markers and *ArgM-TMP* for temporal markers [31]. AUTOMATE enriches SRL with the identification of word connectors specified in Section 3,

in order to recognize LTL relations. Specifically, to detect conditions, AUTOMATE selects roles labeled as ArgM-ADV (adverbial modifiers) or ArgM-DIS (discourse Markers) and considers only the ones containing the words *if/otherwise* or *in case*, thus respecting the defined language constraints. For detecting temporal relations, AUTOMATE selects spans of text labeled as ArgM-TMP and containing words *repeat/until* for loops, or *then* and *once* for sequences.

In order to identify tokens playing the role of *agents*, *objects* or *instruments*, AUTOMATE uses a different strategy for *robot-as-agent* and *surgeon-as-agent* scenarios. In fact, in the first scenario, the robotic arm or instrument plays the role of subject, and thus Arg0 label corresponds to the agent, while the *object* plays the role of a *proto-patient*[31], thus corresponding to Arg1 label. For instance, in the tissue retraction text, the sentence:

The camera selects a point on the tissue if it is far from APs

is labeled as:

[Arg0: The camera] [V: selects] [Arg1: a point on the tissue] [ArgM-ADV: if it is far from APs].

Then, given the *select* action, the agent is the *camera* (Arg0) and the object is *the point on the tissue* (Arg1).

When the surgeon is the agent, the situation is more complex because the instrument/arm is not the subject (Arg0) and may occur in spans of text corresponding to other semantic roles. In this case, AUTOMATE relies on the available list of surgical instruments and searches their mentions among the span of texts labeled as Arg-MNR, Arg2, or Arg3 by SRL. For instance, in the peg transfer scenario, consider the following sentence:

First, the surgeon identifies rings via camera

This is annotated by SRL as:

[ArgM-TMP: First], [Arg0: the surgeon] [V: identifies] [Arg1: rings] [ArgM-MNR: via camera].

The word *camera* is a candidate instrument and it is contained in a span of text labeled as ArgM-MNR by SRL. It is thus recognized as the instrument for the *identify* action.

While instrument/arm tokens can be retrieved from the available list in the setup description, the use of SRL is still necessary. In fact, not all mentions of medical instruments in a sentence refer to the actual usage of an instrument to perform the main action. For instance, Arg1 labels cannot represent agents for the same action they are objects of. Then, SRL is useful to identify the candidate arguments which can actually refer to the instrument, from a semantic perspective. For instance, the sentence:

The surgeon uses the first arm to grasp scissors

is annotated by SRL as:

[Arg0: The surgeon] uses the first arm to [V: grasp] [Arg1: scissors],

that is, *scissors* is correctly recognized as the *thing grasped* (Arg1 of *grasp*), hence it is discarded and *the first arm* (which is still in the list of possible agents) is marked as the actual agent.

Finally, benchmark texts also use pronouns (e.g., *it*) to avoid word repetitions, as a common practice in natural language. AUTOMATE performs the co-reference resolution by mapping each pronoun to the immediately previous semantic role labeled in the same way. While this approach is effective in our experiments, some deeper co-reference resolution systems, e.g., [32], may be needed in more complex scenarios.

5.3 From semantic roles to LTL templates

The output of SRL highlights relevant semantic information about task knowledge, including agents, objects, and temporal-causal relations. This information is then automatically translated to LTL templates, directly mapping temporal-causal roles to logical operators with static rules. Specifically, *if/otherwise* statements are mapped to logical implications ($\rightarrow$), *then/once* are mapped to next ($\circ$) operator and *until/repeat* to until (U) operator.

The following sentences from the texts of our benchmark tasks clarify the process:

[**SEN-1**] In case the point is close to first arm, the point is reached by first arm; otherwise, the second arm reaches the point.
[**SEN-2**] Then, the gripper grasps the tissue.
[**SEN-3**] The arm lifts the tissue until a maximum height is reached, or maximum force is reached, or the ROI is visible.

The respective output of SRL is:

[ArgM-ADV: In case the point is close to first arm,] [Arg1: the point] is [V: reached] [Arg0: by first arm;] [ArgM-ADV: otherwise] [Arg0: the second arm] [V: reaches] [Arg1: the point.]

[ArgM-TMP: Then,] [Arg0: the gripper] [V: grasps] [Arg1: the tissue.]

[Arg0: The arm] [V: lifts] [Arg1: the tissue] [ArgM-TMP: until a maximum height is reached, or maximum force is reached, or the ROI is visible.]

After the previous stage, main actions are represented as predicates *verb(agent, object, other_roles)*, according to the typical formalism of action languages[2]:

reach(the first arm, the point, ADV: in case the point is close to first arm)
reach(the second arm, a point on the tissue, ADV:otherwise)

grasp(first arm, the tissue, TMP:then)

raise(first arm, the tissue, TMP:until a maximum height is reached, or maximum force is reached, or the ROI is visible).

Temporal and causal roles are then directly mapped to corresponding LTL relations by means of logical operators, obtaining LTL templates. For the above example, the output is:

[**LTL-1a**] reach(first arm, a point on the tissue) $\leftarrow$ the point is close to first arm
[**LTL-1b**] reach(second arm, a point on the tissue) $\leftarrow \neg$ the point is close to first arm
[**LTL-2**] $\circ$ grasp(first arm, tissue)
[**LTL-3**] raise(first arm, tissue) U (maximum height is reached $\vee$ maximum force is reached $\vee$ the ROI is visible)

where $\vee$ is the logical *disjunction* and $\neg$ is the logical *negation*. The meaning of underlined parts will be clarified in the next section.

This section concludes the automatic part of AUTOMATE and allows us to reply positively to our research question **RQ1**. In fact, in Section 3 we have started from highly specialized and expressive surgical texts and defined reasonable and loose language constraints based on a domain-specific systematic analysis. This has allowed us to generate texts in Section 4, which preserve most of the expressiveness of natural language, e.g., different verb tenses, the use of synonyms and pronouns, active and passive forms, and richness of semantic roles. AUTOMATE is then able to retrieve LTL templates that correctly express the causal and temporal flow of actions, enucleating relevant procedural information from text.

5.4 From LTL templates to an executable logic program

LTL templates extracted by AUTOMATE must be translated to the syntax of a specific logic program, in order to be effectively used for robotic task planning.

A logic program (based on an action language as, e.g., PDDL) represents a domain of interest with a *signature* (alphabet), defining main variables and predicates of variables (*atoms*), and a set of *axioms* encoding causal and temporal relations between atoms. As an example, consider the sentence from the above section:

The first arm raises the tissue until maximum height is reached [...]

Atoms are used to represent the action predicate raise(first arm, tissue) and the (task-independent) concept *maximum height is reached*, e.g., as `reached(max_height)`.

Atoms for actions can be automatically retrieved from the final step of AUTOMATE (see previous section), where predicates in the form *verb(agent, object)* are automatically built. The only further required step is to map instances of agents to a more general Agent variable, i.e., lifting `raise(first arm, tissue)` to `raise(Agent, tissue)`, in order to build generic LTL rules. This step is performed automatically, by mapping all instrument occurrences in AUTOMATE predicates to the unique variable name.

The atom `reached(max_height)`, as well as parts underlined in statements **LTL1-a** to **LTL-3**, represent *environmental concepts* which are less trivial to represent, hence still require handcrafting from the programmer. However, typically these concepts are not *task-specific*, but they represent at most *domain-specific commonsense* information. For several scenarios of interest, ontologies encoding commonsense concepts are available, e.g., [26] for surgery, [33] for rehabilitation and [34] for robotics and automation. Thus, it is possible to reuse and slightly extend already existing ontologies [35] to define missing atoms in LTL rules. For instance, `reached(max_height)` atom in the previous example can be encoded exploiting the already available *PositionPoint* class in [34].

Finally, as mentioned in the previous section, the mapping of LTL operators to their specific logic programming encoding is performed automatically. The above sentence can then be translated to a statement in logic programming:

$$\text{lift(arm, tissue)}\ U\ \text{reached(max\_force)} \quad (1)$$

This section replies to research question **RQ2**. AUTOMATE retrieves LTL templates that have some missing information, related to commonsense knowledge about the domain of interest. Thus, in order to bridge the gap with actual robotic implementation, it is necessary to retrieve such knowledge from existing domain-specific ontologies, or slightly extend them in case of very specialized and unconventional scenarios.

6 Experimental evaluation

In this section, we evaluate the quality and utility of the procedural knowledge extracted from texts, addressing **RQ3**. In more detail, we verify that the extracted task knowledge is correct and general enough to compute suitable plans for

[2] In the third sentence, notice that *lift* and *raise* are automatically identified as synonyms by WordNet's synsets, and mapped to a single action (arbitrarily chosen as *raise*).

our benchmark robotic tasks, given different initial environmental contexts. This requires:

1. implementation of LTL templates into a specific logic programming language;
2. implementation of low-level routines for robotic motion planning, control, and perception;
3. a simulated environment to replicate the benchmark tasks and the robot.

To address the first requirement, we implement LTL templates in the formalism of Answer Set Programming (ASP), a state-of-the-art logic programming paradigm for autonomous agents [36]. We adopt Clingo 5 [37] software tool for ASP representation and solving (i.e., plan computation). For the second and third requirements, we use the framework for integrated planning and execution of surgical robotic tasks proposed in [25, 38]. This includes a simulated version of the dVRK, the calibrated [39] surgical camera for perception, and the realistic emulation of peg transfer and tissue retraction scenarios. More specifically, we use a CoppeliaSim[3] environment for peg transfer, which involves rigid manipulation; we then use finite-element simulation in Sofa[4] for tissue retraction.

We evaluate the quality of the extracted task specifications in terms of *planning success* and *planning computational performance*.

The planning success measures the percentage of successful generation of task plans in a set of 100 random environmental contexts, corresponding to different workflows of execution, to assess the generality of extracted task knowledge. Contexts differ in the initial location and number of rings for peg transfer, and the locations of ROI and APs for tissue retraction.

The computational performance is calculated as the time required by Clingo to compute a plan, given some initial environmental context. We evaluate this metric as the complexity of the planning problem increases, as explained in the following sections.

For both metrics, we compare the performance of the automatically extracted LTL templates[5] against the ASP task description written by an expert programmer with full knowledge about the domains of interest. There are differences between hand-written ASP programs and LTL templates extracted from procedural texts. In fact, hand-written programs typically encode task knowledge as preconditions and effects of actions, i.e., axioms connecting environmental features to actions, as in standard action languages such as PDDL [2]. Instead, procedural texts typically contain information about the causal/temporal flow of actions. For instance, with reference to the *reach* and *grasp* actions reported in sentences **SEN-1** to **SEN-2** in Section 5.3, their mutual causal and temporal relation is expressed in classical ASP in terms of pre- and post-conditions as follows:

$$\begin{aligned} &\text{at(Agent, tissue, t)} \ :- \text{reach(Agent, tissue, t} - 1) \,. \\ &\text{grasp(Agent, tissue, t)} \ :- \text{at(Agent, tissue, t)} \,. \end{aligned} \tag{2}$$

where :- is the logical implication ($\leftarrow$), t is a variable representing a discrete time step, and at(Agent, tissue, t) is an environmental feature representing the location of an arm with respect to an object, as an effect of *reach* action. We make ASP encodings available in the linked repository.

All experiments are performed in a simulated environment, using a PC with a 2.6 GHz Intel Core i7-6700HQ CPU (4 cores/8 threads) and 16 GB RAM.

6.1 Peg transfer

Below, results for the peg transfer domain are presented.

6.1.1 Planning success

Domain variables which influence the workflow of execution, hence are relevant for assessing planning success, are:

- the number of rings, affecting the number of required actions to complete the task successfully;
- initial placement of the rings on grey pegs, which requires extraction before bringing them to the correct pegs;
- the relative positions of the rings with respect to the robotic arms, affecting reachability conditions and thus possibly requiring transfer between arms before placement on pegs.

Hence, we generate 100 random scenarios as follows:

- 19 scenarios present only 1 ring, 30 scenarios 2 rings, 22 scenarios 3 rings, and 29 scenarios 4 rings[6]
- 84/100 scenarios present at least one ring on a grey peg, so they require extraction;
- 80/100 scenarios require transferring of rings between arms.

[3] https://www.coppeliarobotics.com/

[4] https://www.sofa-framework.org/

[5] AUTOMATE pipeline is able to extract the same LTL relations from *surgeon-as-agent* and *robot-as-agent* texts.

[6] The maximum number of rings in the scene is set to 4 as of FLS specifications [22].

The task is considered successful when all rings are placed on the same-colored pegs. In all scenarios, 100% success rate is achieved by both LTL templates extracted by AUTOMATE and the hand-written ASP encoding.

6.1.2 Computational performance

The complexity of the planning problem depends on the number of rings and actions to be executed (e.g., whether the rings require extraction or not).

In Fig. 3a, we show the ratio between the planning time with the ASP program extracted from the text and the expert-written one, in the 100 random scenarios considered above (sorted by plan length). The ratio decreases significantly as the plan size increases, meaning that LTL templates encode task knowledge more efficiently, enhancing scalability to more complex task instances.

This is even more evident in Figure 3b, showing the planning time for both ASP programs against the plan length. As the plan length increases, the computational performance of the ASP program extracted from text scales linearly with the length of the plan, thus significantly better than the hand-written program with quadratic progression. This happens because of the different ASP representations, with the classical hand-written ASP formalization having more axioms as explained at the beginning of Section 6, which require more computational effort from Clingo.

Notice that the two ASP programs generate plans with different lengths, though under the same initial configurations. This depends on a slightly different action representation. For instance, in the text description, there are actions as *selecting a target ring with camera* which are captured by SRL and then converted to LTL/ASP predicates. However, such actions are not properly moving actions, so they do not affect the workflow of execution. Hence they are not encoded by the expert writing the ASP program from scratch.

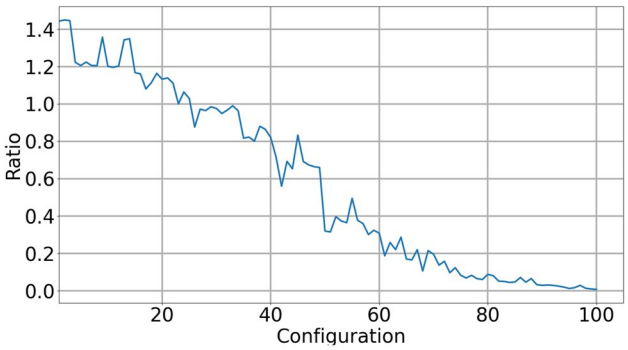


Fig. 3 **On top**, the ratio between planning times with ASP program from text and hand-written ASP program, for 100 random initial configurations sorted by plan length. **In the bottom**, mean and standard planning times for the two ASP programs vs. plan length (mean ± std deviation are reported for plans with the same size). In the box, focus on the results for LTL templates extracted by AUTOMATE (same units as the main plot) is shown

6.2 Tissue retraction

For tissue retraction, we assume that the tissue may be grasped from a discrete set of points, obtained as follows:

1. the rectangular tissue flap is discretized as a $N \times N$ grid;
2. candidate grasping points are centroids of cells in the grid.

At the ASP level, a variable for the candidate grasping point is added, instead of the generic tissue, with a unique identifier for each point in $\{1, ..., N^2\}$. Grasping points are chosen depending on the distance from APs.

6.2.1 Planning success

Variables that affect the workflow of execution are:

- the initial position of the ROI, since grasping and pulling the tissue may not be sufficient to expose it and horizontal folding may be needed;
- a different robotic arm may be needed to grasp the tissue, depending on the reachability of the chosen grasping point.

Hence, we generate random contexts with fixed $N = 5$ grid discretization, specifically 35/100 requiring re-planning and 67/100 requiring usage of the first arm. The task is considered to be successfully executed if the final ROI exposure percentage is $\geq 70\%$. When the hand-written ASP program is implemented for task planning, the planning success rate is 98%, against 94% with the ASP program extracted from text. However, the mean and standard deviation of ROI

exposure is higher with LTL templates (97.47% ± 6.90% vs. 92.26% ± 9.53%). Thus, overall planning success is not significantly different.

6.2.2 Computational performance

The planning time with Clingo depends mainly on the number of candidate grasping points, i.e., the grid discretization parameter N [24]. Hence, we consider different $N \times N$ grid discretizations of the tissue flap, with $N \in \{5, ..., 15\}$, and randomize 20 different locations of APs and ROI for each of them.

In Fig. 4a we show the ratio between planning times obtained with the ASP program extracted from the text and the hand-written one. The ratio is always < 1, meaning that ASP axioms extracted from the text are slightly more efficient than hand-written ones. The ratio does not significantly vary for different tissue discretizations, while the absolute discrepancy between planning times for the two ASP programs increases (Fig. 4b). Thus, extracted LTL templates are still slightly more efficient for the ASP solver.

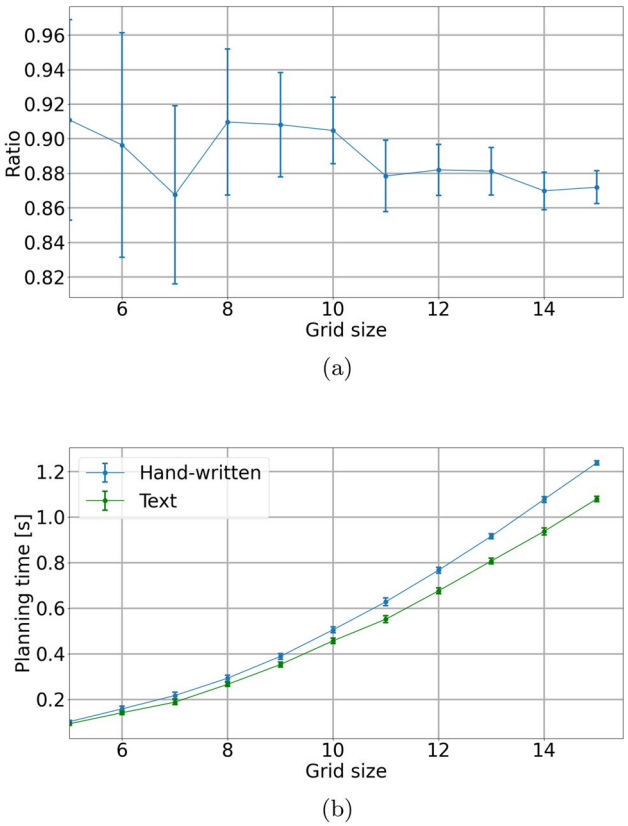


Fig. 4 **On top**, the ratio (mean ± standard deviation) between planning times with ASP program from text and hand-written ASP program for tissue retraction task, for different size N of grid discretization. **In the bottom**, mean and standard planning times for the two ASP programs vs. grid size

6.3 Evaluation considerations

Thanks to the above results, we can positively answer research question **RQ3**: combining LTL templates extracted from AUTOMATE with minimal domain-dependent commonsense knowledge, it is possible to implement more compact computationally efficient logic programs representing task knowledge, preserving the rate of successful planning instances.

7 Conclusion

In this paper, we have presented AUTOMATE, an automatic pipeline for procedural robotic task knowledge extraction from specialized texts. Our methodology tackles the complex problem of identifying LTL templates in expert-written descriptions of tasks, combining automatic procedural sentence classification, POS tagging, SRL, and information filtering rules. Unlike state-of-the-art approaches, AUTOMATE does not rely on oversimplifying language constraints and assumptions. Instead, we consider the very challenging surgical robotic domain, and define the required expressiveness of input texts, based on the analysis of surgical language variability in a publicly available dataset of procedural annotations and manuals. Moreover, AUTOMATE is robust concerning several aspects of natural language variability, never fully addressed in previous research. First, it can identify all main LTL operators, e.g., $\circ$ and U, as well as standard logical connectors. Secondly, it can deal with different verbal configurations (i.e., active and passive forms and modal verbs) and perspectives in writing (i.e., texts written from the point of view of the human or the robot). Finally, we empirically show that to bridge the gap between procedural texts and effective implementation of extracted knowledge, thus obtaining logic programs for actual robotic task automation, low-level commonsense knowledge is required, that is often a-priori coded or contained in domain ontologies. In the context of two simulated benchmark surgical training tasks, we have shown that automatically extracted task knowledge is sufficient to successfully complete multiple random task instances, outperforming logic programs written by expert programmers in terms of computational efficiency. AUTOMATE also offers a unique advantage for robotic programmers, since it provides them with clear LTL specifications about a task and domain of interest, which only requires to be translated into a specific logic program. This is fundamental, especially in complex scenarios, where programmers have limited domain knowledge. Moreover, AUTOMATE can be easily extended to other domains of interest, after defining suitable language constraints accordingly.

In conclusion, we can positively answer **RQ1**: extracting LTL templates from specialized texts is feasible, without relying on oversimplifying language restrictions, but only on a systematic linguistic analysis of the domain. For **RQ2**,

we state that translating LTL templates to an executable logic program is possible, under the assumption that minimal task-independent commonsense knowledge about the domain is available, e.g., from existing domain ontologies. Finally, results show that the LTL program extracted from text outperforms a handcrafted one in terms of computational time while guaranteeing the same success rate, thus positively answering **RQ3**. This paper opens several future work directions. One fundamental consideration is about the *completeness* of task knowledge available in specialized texts. In fact, especially in complex scenarios such as surgical robotics, only nominal procedures, and the most common variants are typically described. Thus, it is important to integrate extracted LTL templates, with task knowledge inferred from available datasets of execution, e.g., employing unsupervised action segmentation [40] and inductive learning to discover task specifications [41]. In this way, it is possible to guarantee that the autonomous system can deal with the highest number of unexpected events. Moreover, to apply our pipeline to more challenging surgical procedures, we will adopt NLP domain adaptation techniques, following recent research trends [42, 43] to deal with highly technical expressions in texts. Finally, we plan to validate AUTOMATE in different domains and to bring extracted logic programs on a real robotic setup, to integrate task planning with the challenges of perception and motion control.

Funding Open access funding provided by Università degli Studi di Verona within the CRUI-CARE Agreement. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 742671 "ARS").

Data availability We make the autonomous framework publicly available at https://gitlab.com/altairLab/AUTOMATE

Declarations

Conflicts of interest All the authors declare that they have no conflict of interest.

Ethical approval This article does not contain studies with human participants or animals.

Informed consent Statement of informed consent is not applicable since the manuscript does not contain any patient data.

References

1. Proposal for a regulation of the European parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain Union legislative acts (2021).https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52021PC0206
2. Haslum P, Lipovetzky N, Magazzeni D, Muise C (2019) An introduction to the planning domain definition language. Synth Lect Artif Intell Mach Learn 13(2):1–187
3. Apt KR (1990) Logic programming. Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), 493–574
4. Park H, Motahari Nezhad HR (2018) Learning procedures from text: Codifying how-to procedures in deep neural networks. Comp Proc Web Conf 2018:351–358
5. Hsiung E, Mehta H, Chu J, Liu X, Patel R, Tellex S, Konidaris G (2022) Generalizing to new domains by mapping natural language to lifted ltl. In: 2022 International Conference on Robotics and Automation (ICRA), pp. 3624–3630. IEEE
6. Pnueli A (1977) The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science (1977), pp. 46 – 57. IEEE
7. Yang G-Z, Cambias J, Cleary K, Daimler E, Drake J, Dupont PE, Hata N, Kazanzides P, Martel S, Patel RV et al (2017) Medical robotics-regulatory, ethical, and legal considerations for increasing levels of autonomy. Sci Robot 2(4):8638
8. Mujtaba D, Mahapatra N (2019) Recent trends in natural language understanding for procedural knowledge. In: 2019 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 420–424
9. Brunello A, Montanari A, Reynolds, M (2019) Synthesis of LTL formulas from natural language texts: State of the art and research directions. In: 26th International Symposium on Temporal Representation and Reasoning, TIME 2019, October 16-19, 2019. LIPIcs, vol. 147, pp. 17–11719. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Málaga, Spain
10. Wambsganss T, Fromm H (2019) Mining user-generated repair instructions from automotive web communities. In: Bui T (ed) 52nd Hawaii International Conference on System Sciences, HICSS 2019. ScholarSpace, Grand Wailea, Maui, Hawaii, USA, pp 1–10
11. Qian C, Wen L, Kumar A, Lin L, Lin L, Zong Z, Li S, Wang J (2020) An approach for process model extraction by multi-grained text classification. In: Dustdar S, Yu E, Salinesi C, Rieu D, Pant V (eds) Advanced Information Systems Engineering. Springer, Cham, pp 268–282
12. Nabizadeh N, Wersing H, Kolossa D (2021) Leveraging inter-step dependencies for information extraction from procedural task instructions . In: Text, Speech, and Dialogue - 24th International Conference, TSD, Proceedings. Lecture Notes in Computer Science , vol. 12848 , pp. 341 – 353. Springer, Olomouc, Czech Republic
13. Jiang Y, Zaporojets K, Deleu J, Demeester T, Develder C (2023) Cookdial: a dataset for task-oriented dialogs grounded in procedural documents. Appl Intell 53(4):4748–4766
14. Zhong B, Xing X, Luo H, Zhou Q, Li H, Rose TM, Fang W (2020) Deep learning-based extraction of construction procedural constraints from construction regulations. Adv Eng Inf 43:101003
15. Wang X, El-Gohary N (2023) Deep learning-based relation extraction and knowledge graph-based representation of construction safety requirements. Autom Const 147:104696

16. Chambers AJ, Stringfellow AM, Luo BB, Underwood SJ, Allard TG, Johnston IA, Brockman S, Shing L, Wollaber AB, VanDam C (2020) Automated business process discovery from unstructured natural-language documents. In: Business Process Management Workshops - BPM 2020 International Workshops. Lecture Notes in Business Information Processing , vol. 397, pp. 232–243. Springer, Seville, Spain
17. Ning Q, Zhou B, Feng Z, Peng H, Roth D (2018) Cogcomptime: A tool for understanding time in natural language . In: Blanco , E. , Lu , W. (eds.) Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018 , pp. 72 – 77 . Association for Computational Linguistics, Belgium
18. Bombieri M, Rospocher M, Dall'Alba D, Fiorini P (2021) Automatic detection of procedural knowledge in robotic-assisted surgical texts. Int J Comput Assist Radiol Surg 16(8):1287–1295
19. Fellbau C (1998) Wordnet: An electronic lexical database
20. Bentivogli L, Bocco A, Pianta E (2004) Archiwordnet: integrating wordnet with domain-specific knowledge. In: Proceedings of the 2nd International Global Wordnet Conference, pp. 39–47
21. Kazanzides P, Chen Z, Deguet A, Fischer GS, Taylor RH, DiMaio SP (2014) An open-source research kit for the da vinci surgical system. In: 2014 IEEE International Conference on Robotics and Automation, ICRA 2014, May 31 - June 7, 2014 , pp. 6434 – 6439. IEEE , Hong Kong, China
22. Soper NJ, Fried GM (2008) The fundamentals of laparoscopic surgery: its time has come. Bull Am Coll Surg 93(9):30–32
23. Nagy TD, Haidegger TP (2021) Towards standard approaches for the evaluation of autonomous surgical subtask execution. In: 2021 IEEE 25th International Conference on Intelligent Engineering Systems (INES), pp. 67–74. IEEE
24. Meli D, Tagliabue E, Dall'Alba D, Fiorini P (2021) Autonomous tissue retraction with a biomechanically informed logic based framework. In: 2021 International Symposium on Medical Robotics (ISMR). IEEE, Atlanta, GA, pp 1–7. https://doi.org/10.1109/ISMR48346.2021.9661573
25. Tagliabue E, Meli D, Dall'alba D, Fiorini P (2022) Deliberation in autonomous robotic surgery: a framework for handling anatomical uncertainty. In: Proceedings-IEEE International Conference on Robotics and Automation, pp. 11080–11086
26. Gibaud B, Forestier G, Feldmann C, Ferrigno G, Gonçalves P, Haidegger T, Julliard C, Kati D, Kenngott H, Maier-Hein L et al (2018) Toward a standard ontology of surgical process models. Int J Comput Assist Radiol Surg. 13(9):1397–1408
27. Fiorini SR, Bermejo-Alonso J, Gonçalves P, De Freitas EP, Alarcos AO, Olszewska JI, Prestes E, Schlenoff C, Ragavan SV, Redfield S et al (2017) A suite of ontologies for robotics and automation [industrial activities]. IEEE Robot Autom Mag 24(1):8–11
28. Shi P, Lin J (2019) Simple BERT models for relation extraction and semantic role labeling. CoRR abs/1904.05255
29. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)
30. Bird S (2006) NLTK: The Natural Language Toolkit. In: Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions , pp. 69 – 72 . Association for Computational Linguistics , Sydney, Australia
31. Palmer M, Kingsbury PR, Gildea D (2005) The proposition bank: An annotated corpus of semantic roles. Comput Linguistics 31(1):71–106
32. Dobrovolskii V (2021) Word-level coreference resolution . In: Moens , M. , Huang , X. , Specia , L. , Yih , S.W. (eds.) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event, 7-11 November, 2021 , pp. 7670 – 7675 . Association for Computational Linguistics , Punta Cana, Dominican Republic
33. Dogmus Z, Gezici G, Patoglu V, Erdem E (2012) Developing and maintaining an ontology for rehabilitation robotics. In: KEOD , pp. 389 – 395
34. Schlenoff C, Prestes E, Madhavan R, Goncalves P, Li H, Balakirsky S, Kramer T, Miguelanez E (2012) An ieee standard ontology for robotics and automation. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems , pp. 1337 – 1342. IEEE
35. Guerram, T, Mellal N (2018) A domain independent approach for ontology semantic enrichment. Computer Science & Information Technology, 13–19
36. Meli D, Nakawala H, Fiorini P (2023) Logic programming for deliberative robotic task planning. Artif Intell Rev 56:9011–9049
37. Gebser M, Kaminski R, Kaufmann B, Ostrowski M, Schaub T, Wanko P (2016) Theory solving made easy with clingo 5. In: Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
38. Ginesi M, Meli D, Roberti A, Sansonetto N, Fiorini P (2020) Autonomous task planning and situation awareness in robotic surgery. In: IEEE/RSJ International Conference on Intelligent Robots and Systems , pp. 3144 – 3150. IEEE
39. Roberti A, Piccinelli N, Meli D, Muradore R, Fiorini P (2020) Improving rigid 3-d calibration for robotic surgery. IEEE Trans Med Robot Bionics 2(4):569–573
40. Meli D, Fiorini P (2021) Unsupervised identification of surgical robotic actions from small non-homogeneous datasets. IEEE Robot Autom Lett 6(4):8205–8212
41. Meli D, Sridharan M, Fiorini P (2021) Inductive learning of answer set programs for autonomous surgical task planning. Mach Learn 110:1739–1763
42. Bombieri M, Rospocher M, Ponzetto SP, Fiorini P (2022) The Robotic Surgery Procedural Framebank . In: Proceedings of the Thirteenth International Conference on Language Resources and Evaluation (LREC 2022). European Language Resources Association (ELRA), Marseille, France
43. Bombieri M, Rospocher M, Ponzetto SP, Fiorini P (2023) Machine understanding surgical actions from intervention procedure textbooks. Comput Biol Med 152:106415