From High to Low Level and Vice-Versa: A New Language for the Translation between Abstraction Levels in Robot Control Architectures

José Carlos González, Javier García, Raquel Fuentetaja, Angel García-Olaya, Fernando Fernández
Universidad Carlos III de Madrid
Avenida de la Universidad 30, 28911 Leganes, Madrid, Spain
fjgpolo@inf.uc3m.es

***Abstract*— The use of Planning, Execution and Monitoring architectures to control robotic platforms is becoming very popular. In most cases these architectures provide knowledge at two different levels of abstraction: high-level (deliberative planning), and low-level (robot sensing and reactive behaviours). Therefore, the translation between these two levels of abstraction is required to solve real use cases. Typically such translations are written in the source code by experts who know the software. Furthermore, if these translations or the robotic platform change, it is required to resort such experts again for editing the source code to incorporate all these changes and, in the worst case, to recompile the entire software architecture. It would be useful if such translations could be defined in a declarative way, so that they can be easily edited (even by non-experts) and without modifying the modules of the control architecture, which should be able to process such formal description. For this reason, we contribute with a language for the description of translations from High to Low and from Low to High abstraction levels when designing robotic planning tasks.**

I. Introduction

Automated Planning (AP) has been successfully applied to different real-world problems, such as robot control [2]. To take advantage of the benefits of AP in robot control, the use of Planning, Execution and Monitoring architectures is becoming very popular. An example of such architecture is PELEA [2], [3] which performs a cycle of constructing an initial plan to achieve certain goals, monitoring the plan execution, analyzing deviations from the original plan, re-planning when unexpected situations are found, and executing the new plan. This cycle requires the control architecture to sense external information (velocities, distances, forces), translate this low-level information to a high level of abstraction interpretable by the planning system, and also translate the high-level actions of a plan to low-level actions interpretable by the robot. Typically, these translations are written by experts in the source code of the control architecture itself, in an *ad hoc* manner for each particular robot platform and task planning system. In this way, including modifications requires, on one hand, to resort to experts who know the software architecture and, on the other hand, to rewrite the source code, recompiling the entire software architecture in the worst case. However, ideally, the source code should not be altered by these modifications, which should be able to be carried out even by users with little knowledge about planning and robotics.

For this reason, in this paper, we contribute with a declarative language for the description of such translations between abstraction levels for robotic planning tasks. The formal descriptions can be processed by the software architecture, which favours its management, as well as the update of the architecture with new modifications incorporated to the robot (e.g., new set of low-level actions or new sensor information).

II. PELEA: Robotic Control Architecture

Figure 1 shows a representation of the main modules of the PELEA architecture.

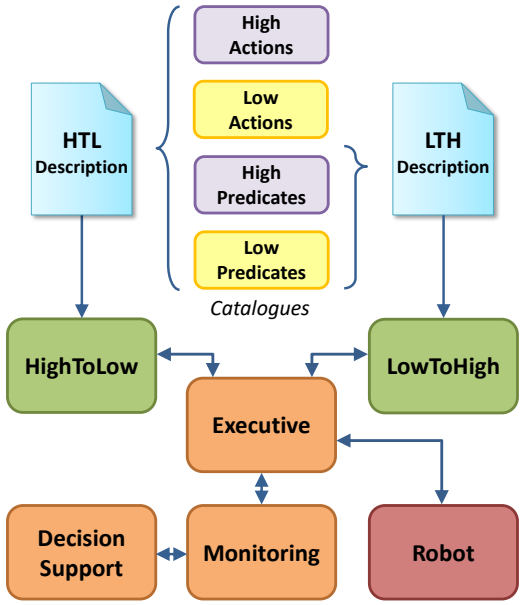

Fig. 1. PELEA architecture.

Further details of PELEA are provided in the reference section [2], [1], [3]. Here, we will focus only in the way in which the translations between different levels of abstraction are carried out in the architecture. For this purpose, the user is provided with several catalogues that separate high-level *concepts* from low-level ones. *High Actions* and *High Predicates* catalogues in Figure 1 represent respectively actions and predicates in the description of the planning domain. Meanwhile, *Low Actions* and *Low Predicates* define respectively low-level instructions interpretable by the robot, and low predicates that can be both external information describing the current state of the world or raw sensor data. Therefore, the translations consist in relating *High* and *Low* concepts from these catalogues.

As an example of translation from high to low, let us assume we have the high-level action *SAY (speech, behavior)* in *High Actions* catalogue, and let us also assume we have several low level instructions in *Low Actions* catalogue. These low actions are *say*, used to make the robot speak; *show-video*, used to show a video related to what the robot is saying; and *insert-subtitle*, used to show on a screen what the robot is saying. High and low actions from these catalogues can be related differently depending on the behavior desired by the user. For instance, it may be desired that *SAY* is translated into *say*, and *show-video*; or it may be desired that it is translated into *say*, and *insert-subtitle*; or, finally, this translation may depend on the value of the parameter *behavior* of *SAY*. Regardless the translation desired by the user, they are written in the *HTL Description* file in Figure 1 using the proposed language as described in Section III.

The translation from low to high works in a similar way. In this case, let us suppose we have the predicates *can_continue* and *detected_event(person)* in *High predicates* catalogue, and we have the predicate *person* in *Low predicates* catalogue that tells us whether the person the robot speaks to is detected. A possible translation could be removing the predicates *can_continue* and *detected_event(person)* from the current state each time *person* is false. This translation is written in the *LTH Description* file in Figure III also using the proposed language as described in Section III.

Both files, *HTL Description* and *LTH Description* are used by the green modules in Figure 1 to perform the translations. Therefore, what we propose is to separate the concepts of high and low in different catalogues so that even non-expert users can be able to relate them in description files depending on the desired behavior for the robot. The declarative language proposed in Section III is used to build such files.

III. EBNF Description of the Language

The EBNF description of the syntax for high to low translations is as follows:

```
<high-to-low> ::= <high-to-low-statement>
                  {<high-to-low-statement>}
<high-to-low-statement> ::= <high-statement>
                            {<high-statement>}
                             <low-statement>
                            {<low-statement>}
<high-statement> ::=
     High: <string> ([<param-list>]) [, <cond-expr>]
<low-statement> ::= Lows: <string> ([<param-list>])
                         {<string> ([<param-list>])}
<param-list> ::= <string> {, <string>}
<cond-expr> ::= <operand0> <bool-op> <operand1>
<operand0> ::= <string> | <cond-expr>
<operand1> ::= <number> | <boolean> | <string>
<bool-op> ::= and | or | == | != | > | >= | < | <=
<boolean> ::= true | false
<number> ::= Any integer literal
<string> ::= Any string literal
```

For simplicity, this and following EBNF syntax descriptions do not address issues as comment conventions and the use of "white space" to delimit tokens.

If we wanted to translate the high level action *SAY* described in Section II using the proposed syntax, the result would be the following:

```
High: say(speech,behavior),$behavior==show_video
Lows: say(speech)
      show_video()
High: say(speech,behavior),$behavior==insert_subtitle
Lows: say(speech)
      insert_subtitle()
```

The first entry applies when `behaviour` is `show_video`, while the second entry applies when `behaviour` is `insert_subtitle`. As can be seen in the syntax description, more complex conditional expressions (involving `and` and `or` operators) can be built indicating whether the translation is applicable or not.

The EBNF description for low to high translations is:

```
<low-to-high> ::= <low-to-high-statement>
                  {<low-to-high-statement>}
<low-to-high-statement> ::= If: <cond-expr>
                                <state-operator>
                                {<state-operator>}
<state-operator> ::= add(<param>) | delete(<param>) |
increase(<param> <number>) |
decrease(<param> <number>)
<param> ::= <string> [<string>]
```

Therefore, the translation from low to high described in Section II using this syntax would be as follows:

```
If: $person==false
delete(detected_event person)
delete(can_continue)
```

In this case, if the person the robot speaks to is not detected, two high-level predicates in current state of PELEA are removed: *detected_event(person)* and *can_continue*.

IV. Conclusions

The use of the language proposed in Section III is just beginning being used in different robotic platforms [1], [3], and the results found so far are promising. It facilitates to understand, maintain and change the description of the abstractions and provides generality to the control architecture, making it totally independent from the specific domain knowledge. As future work we propose to demonstrate that the use of this language enables useful analysis that one could not achieve using, e.g., a python script.

References

[1] José Carlos González, Fernando Fernández, Ángel García-Olaya, and Raquel Fuentetaja. On the Application of Classical Planning to Real Social Robotic Tasks. In *Proceedings of the 5th Workshop on Planning and Robotics (PlanRob), ICAPS conference*, pages 38–47, Pittsburgh, Pennsylvania, USA, June 2017.

[2] Ezequiel Quintero, Vidal Alcázar, Daniel Borrajo, Juan Fernández-Olivares, Fernando Fernández, and Ángel García Olaya. Autonomous Mobile Robot Control and Learning with the PELEA Architecture. In *Proceedings of the 9th AAAI Conference on Automated Action Planning for Autonomous Mobile Robots (PAMR)*, San Francisco, California, USA, Aug. 2011.

[3] Antonio Jesús Bandera Rubio, Juan Pedro Bandera Rubio, Pablo Bustos, Luis Vicente Calderita, Álvaro Dueñas, Fernando Fernández, Raquel Fuentetaja, Ángel Garía-Olaya, Javier García, José Carlos González, and Christian Reuther. CLARC: a Robotic Architecture for Comprehensive Geriatric Assessment. In *Proceedings of the 17th Workshop of Physical Agents (WAF)*, Málaga, Spain, June 2016.