

APPAREIL: A Tool for Building Automated Program Translators Using Annotated Grammars

Diego Ordóñez Camacho
Université catholique de Louvain, Belgium
diego.ordonez@uclouvain.be

Kim Mens
Université catholique de Louvain, Belgium
kim.mens@uclouvain.be

Abstract—Operations languages are used to write spacecraft operations procedures. The APPAREIL tool automates the process of generating program translators between operations languages, from a specification of their language grammar annotated with extra information. From these annotated grammars the tool automatically produces a partial translator that covers most of the translation. This translator needs to be augmented manually with specific transformations, to deal with the more complicated cases. To get more confidence on the correctness of the translation, the tool offers a control-flow equivalence verification module.

I. INTRODUCTION

Automatically generating language translators, based merely on source and target language specifications, is a complex problem. Restricting ourselves to a given family of domain-specific languages, that have a common semantic basis, significantly simplifies the problem. When viewed from this angle, the problem becomes more syntactic in nature, thus enabling a much higher degree of automation. There exist several families of domain-specific languages where such a common semantic basis exist, and we are focusing on the *Operations Languages* (OLs) used in spacecraft mission planning.

Although different OLs have different syntax, overall they share many features. This is because they all address the same goal of communicating with a spacecraft and they all respect known standards on satellite construction and operation. Programs written in OLs are generally known as spacecraft procedures. They describe a set of instructions that need to be executed by a spacecraft, and therefore must have knowledge about that spacecraft and its behaviour. They need to know how to retrieve information from the spacecraft (telemetry) and how to make the spacecraft execute certain commands (telecommand). This knowledge is typically contained in a database called the Mission Information Base (MIB) [1]. The semantics of OLs is therefore limited essentially to describing the overall flow of control of procedures, organising the way they execute the telecommands and read telemetry data.

The approach we propose to translate between any two operations languages, is a generic technique to semi-automatically derive translators from one OL to another, and to verify control-flow equivalence of the resulting translation, based on the corresponding context-free grammars of those languages annotated with additional semantic information.

II. THE APPAREIL APPROACH

Figure 1 provides a schematic overview of our automated approach to program translation [2], [3], [4], indicating the three different kinds of actors involved. The bottom level

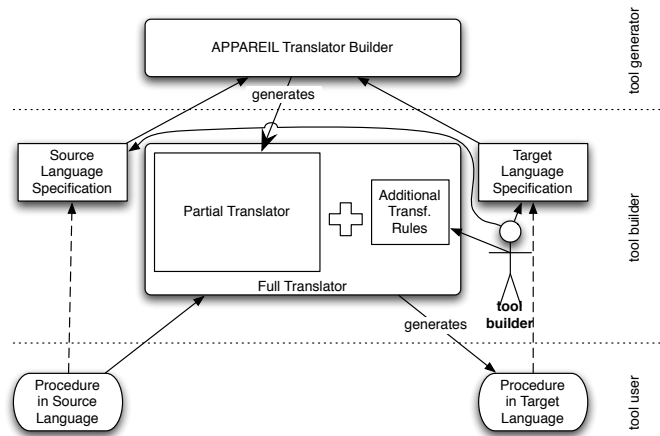


Fig. 1. The APPAREIL Model

represents the end users, who only care about having a program transformer, to which they can feed programs written in their source programming language and which produces an equivalent program in the target language. At the intermediate level, we have the tool builders who provide the end users with a transformation tool for the source and target programming language of their choice. To build such a program transformation tool, they make use of our tool, which is situated at the top level. The tool builders provide the top level tool with a specification of the grammar of both source and target language, tagged with annotations that specify the correspondence between language constructs in the two languages and its control-flow semantics, as illustrated in Figure 2. Using this input, the top level tool then semi-automatically generates a dedicated transformer for translating programs from the source to the target language, and the tool builder has to intervene only to specify how to translate those cases for which no direct equivalence could be stated between productions in the source and target grammars.

The full flow of the APPAREIL process for producing language translators, as depicted in Figure 3, can be described in five main steps. First (1) we start from the OL documen-

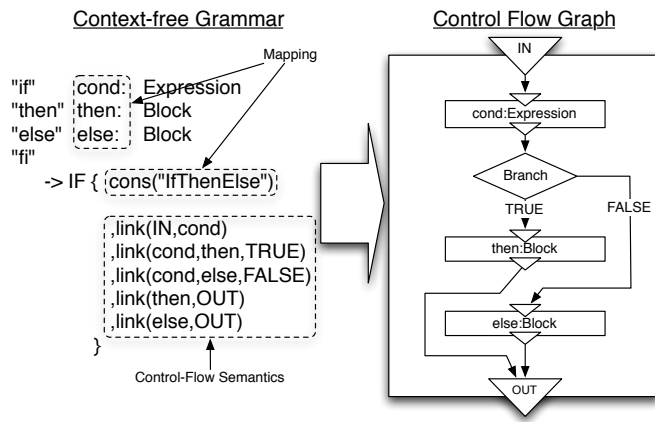


Fig. 2. The APPAREIL Annotations

tation that provide the specifications and definitions of the language. Since often this documentation is not complete or too informal as to be used directly, we need to complete it and correctly structure it. We use the ASF+SDF Meta-Environment [5] as a support tool to design a working SDF grammar for both source and target languages. Next (2), we annotate those grammars with extra annotations defining the mapping and the control-flow semantics [3]. The third step (3) is fully automatic. It takes as input the annotated grammars of both languages, and produce a partial translator for them. Nevertheless, this translator, depending on how similar both languages are, could require a certain amount of extra work to complete with additional transformations. In step (4) we cope with those cases where the nature of the mismatches between languages are such that an automatic transformation cannot be derived only based on annotations. The previous step already provides relevant information, extracted from an analysis of the annotations mapping, signalling the places where mismatches and incompatibilities have been found. We use a dedicated transformations library to compose the additional transformations to include in the translator. Finally (5), we simply use the now complete translator, providing it with the procedures (programs) to translate, and verify the result of this translation with our verification module, that will check for control-flow equivalence between original and translated procedures. This step could be cyclic, especially in cases when some further modifications to the translator need to be made due to errors reported by the verification module. A report from the bisimulation, providing different levels of warnings, can be analysed to decide if the transformation is reasonable enough.

III. INDUSTRIAL RELEVANCE

Automated support for translating procedures between operations languages, in the domain of spacecraft mission planning, is an important issue [3]. There is an industrial demand for such general-purpose tools that can manipulate and translate between procedures in any of the many operations languages that currently exist. The current tendency to strive

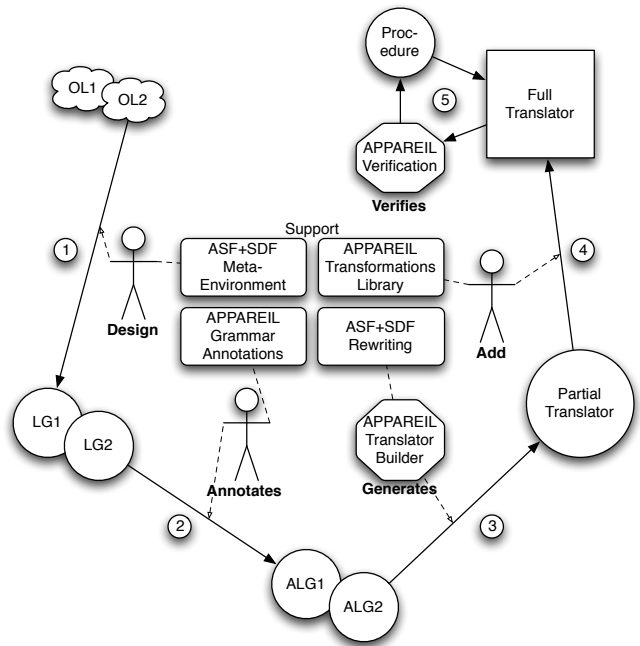


Fig. 3. The APPAREIL Process.

towards a standard operations language strengthens the need for translation tools, not only to translate procedures in old operations languages to the new standard language, but also because there is a need for translating between the two languages competing to become a standard.

ACKNOWLEDGEMENT

This research was supported by Rhea System and the Walloonian Region in Belgium, in the context of the FIRST Europe Objectif 3 research project APPAREIL, and by the MoVES project of the Interuniversity Attraction Poles Programme of the Belgian Science Policy (Belgian State).

REFERENCES

- [1] L. J. Timmermans, T. Zwartbol, B. A. Oving, and A. A. Casteleijn, "From simulations to operations: Developments in test and verification equipment for spacecraft," *DATA Systems In Aerospace*, 2001.
- [2] D. Ordóñez Camacho and K. Mens, "Using annotated grammars for the automated generation of program transformers," in *Ingénierie Dirigée par les Modèles, IDM2007, proceedings*. Toulouse, France: Eds. Antoine Beungard & Marc Pantel, 2007, pp. 7 – 24.
- [3] D. Ordóñez Camacho, K. Mens, J. Cater, and D. Quigley, "Issues and problems in operations language translation," in *SpaceOps*. AIAA, 2008.
- [4] D. Ordóñez Camacho, K. Mens, M. van den Brand, and J. Vinju, "Automated derivation of translators from annotated grammars," *Electronic Notes in Theoretical Computer Science*, vol. 164, Issue 2, pp. 121–137, 2006.
- [5] M. van den Brand, A. van Deursen, J. Heering, H. de Jonge, M. de Jonge., T. Kuipers, P. Klint, L. Moonen, P. Olivier, J. Scheerder, J. Vinju, E. Visser, and J. Visser, "The ASF+SDF Meta-Environment: a component-based language development environment," in *Compiler Construction 2001 (CC 2001)*, ser. LNCS, R. Wilhelm, Ed., vol. 2027. Springer-Verlag, Apr. 2001, pp. 365–370.