Editor Services for MCNP6 in the NEAMS Workbench

Kurt A. Dominesey and Wei Ji

Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY
domink@rpi.edu, jiw2@rpi.edu

INTRODUCTION

MCNP6 (Monte Carlo N Particle code, Version 6) [1] is a popular, general-purpose tool for radiation transport and reactor physics. However, effective manual creation, editing, and inspection of MCNP6 input files is stymied by a lack of editor services (such as content-assist, automatic validation, and rename-refactoring). To address this need, we herein present an implementation of MCNP-specific editor features (developed using Xtext [2]) to be integrated into the NEAMS Workbench [3] via the Language Server Protocol (LSP). This effort supports the broader integration of MCNP6 into the NEAMS Workbench under the ongoing Department of Energy (DOE) Nuclear Energy University Program (NEUP) project: “Integration of high-fidelity Monte Carlo and deterministic transport codes into Workbench”. In particular, this work complements that of [4], as we develop these editor features in parallel with the MCNP parser, serializer, and Application Program Interface (API) sought therein.

BACKGROUND

Xtext is a framework for developing Domain Specific Languages (DSLs), or small, textual, human-readable languages with specific, well-defined use-cases. In the case of MCNP, the domain is modeling Monte Carlo radiation transport problems and the language consists of newline-delimited “cards”, on which positional and keyword arguments are entered. An example of such a card, used to define a plane by the equation $ax + by + cz = d$, is provided in Listing 1.

Listing 1. An MCNP card representing a plane.
1 P 1.0 2.0 3.0 4.0

For this effort, our primary development artifact is an Xtext “grammar” which encodes MCNP’s input specifications in a machine-readable format. Many features are derived automatically from the grammar specification (such as content-assist and references), but may be further customized by the developer. An exemplary grammar rule which could be used to define the card in Listing 1 is given by Listing 2.

Listing 2. An Xtext grammar rule defining a plane card.
Plane:
   name=ID 'P' a=Double b=Double c=Double d=Double

Once a grammar has been defined, the language features can be implemented in an editor. In the present case, we encapsulate our services as a language server, which provides features via the LSP, an open-source standard pioneered by Microsoft [5]. In short, the LSP defines an interface (via JSON Remote Procedure Calls, RPCs) by which clients (editors) and servers communicate to provide users with intelligent language features. As the generation of a language server can be largely automated using Xtext, this artifact is achieved with minimal additional development effort.

IMPLEMENTATION

As discussed previously, creating a formal MCNP grammar specification is the primary thrust of this effort. Unfortunately, MCNP does not readily admit a simple or compact expression of the input specification, usually requiring individual parser rules for each card. Moreover, several constructs, such as macros and the vertical input format, require the generation or reorganization of tokens between the lexing and parsing stages, significantly complicating development.

For these reasons, the MCNP specification discussed herein is preliminary and subject to later expansion and refinement. A summary of the current development progress based on the MCNP6.1 manual [1] is given in Table I. (One may note minor discrepancies between the number of cards per category given below and in the manual, as some cards are split and some are combined within the grammar.)

TABLE I. Status of grammar development.

<table>
<thead>
<tr>
<th>Category</th>
<th>Complete</th>
<th>Total</th>
<th>% Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cells</td>
<td>1</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>Surfaces</td>
<td>37</td>
<td>40</td>
<td>92.50</td>
</tr>
<tr>
<td>Geometry</td>
<td>15</td>
<td>17</td>
<td>88.24</td>
</tr>
<tr>
<td>Physics</td>
<td>20</td>
<td>25</td>
<td>80.00</td>
</tr>
<tr>
<td>Source</td>
<td>10</td>
<td>13</td>
<td>76.92</td>
</tr>
<tr>
<td>Tally</td>
<td>32</td>
<td>36</td>
<td>88.99</td>
</tr>
<tr>
<td>Var. Reduction</td>
<td>20</td>
<td>21</td>
<td>95.24</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>12</td>
<td>15</td>
<td>80.00</td>
</tr>
<tr>
<td>Overall</td>
<td>157</td>
<td>180</td>
<td>87.22</td>
</tr>
</tbody>
</table>

To demonstrate a functional editor integration, we implement our MCNP server as an extension for Visual Studio Code (which we adapt from the work of [6]). Future efforts will seek to integrate our server in the NEAMS Workbench, once the associated LSP client has been finalized. Figures 1 through 3 display the use of several MCNP language features in Visual Studio Code.

All features displayed in these figures represent the default tooling provided by Xtext, as derived from our MCNP grammar. As such, these services are available in all contexts of an MCNP file defined by the grammar, not simply those shown here. More sophisticated features (for example, template suggestions for cards or the validation of custom programmatic requirements) may be developed in later work to address specific areas of user difficulty.
CONCLUSIONS

In summary, we have implemented language services for MCNP6 using Xtext, to be integrated within the NEAMS Workbench via the LSP. Specifically, we have developed a formal input specification, or grammar, supporting approximately 87% of MCNP6.1 cards and demonstrated the use of our server within Visual Studio Code (displaying such features as content-assist, reference resolving, and automatic validation). These utilities alleviate many common difficulties users face when constructing, editing, or viewing MCNP input files. Future work will seek to deploy this language server within the NEAMS Workbench, supporting the broader MCNP integration effort outlined by the DOE NEUP project: “Integration of high-fidelity Monte Carlo and deterministic transport codes into Workbench”, and in complement to the work set forth in [4].

ACKNOWLEDGMENTS

The authors of this paper would like to thank Robert Lefebvre and Forrest Brown for their feedback and counsel throughout the development process. This material is based upon work supported by the Department of Energy Office of Nuclear Energy under Award Number DE-NE0008707.

REFERENCES