SemGuS-Lib Format 1.0

Jinwoo Kim

October 28, 2021

This document defines the current version of the SemGuS format, which is intended to be the standard input and output format for solvers aiming to solve SemGuS problems. The current version of the SemGuS format is designed such that existing solvers supporting the SyGuS format or the SMT-LIB standard will be able to parse SemGuS problems with minimal overhead. As such, the SemGuS format borrows many concepts and language constructs from SyGuS and SMT-LIB. In particular, where the syntax and semantics of constructs are identical, we will refrain from copying the whole definition and instead refer readers to the corresponding sections of the SyGuS and SMT-LIB standard documents, partly in order to keep in sync with updates to the other formats.

This document is currently incomplete; specifically, it is missing a section that details the semantics of each command that we will describe in this document. This will be added in a future version, along with example SemGuS problems.

An instance of a SemGuS problem consists of the following:

1. A base background theory T,
2. A finite set of *term types* that dictate a universe of terms to be used in the synthesis problem, specified as a regular tree grammar (RTG) R
3. *Semantics* for each constructor within the term-type declaration, specified using Constrained Horn Clauses $c_1, c_2, \cdots, c_i$
4. A set of functions $f_1, f_2, \cdots, f_j$ to synthesize,
5. Syntactic constraints for $f_1, f_2, \cdots, f_j$, also specified as a RTG G that defines a *subset* of terms inside the universe of terms defined by R
6. A semantic constraint ϕ that act as the behavioral specification of the set of functions to synthesize, which may be specified over the CHCs $c_1, c_2, \cdots, c_i$, a set of (implicitly) universally quantified variables $x_1, x_2, \cdots, x_k$, and the functions $f_1, f_2, \cdots, f_j$ themselves.

The goal of a SemGuS problem is to find a term $e \in G$ such that $\phi[e/f]$ is true.

1 Syntax

We now introduce the syntax for specifying SemGuS problems. A SemGuS problem $\langle Semgus \rangle$ is defined as a list of commands $\langle Cmd \rangle$s:

$$\langle Semgus \rangle ::= \langle Cmd \rangle^*$$

We now introduce the possible cases that a command $\langle Cmd \rangle$ may take first; then we introduce any required sub-expressions.

1.1 Comments

Comments in SemGuS, like SyGuS and SMT-LIB, are indicated by a leading semicolon. Upon encountering a semicolon, the rest of the input upto the next newline character should be ignored.

1.2 Metadata

SemGuS supports recording metadata related to SemGuS problems (such as problem authors, creation date, expected answers, etc.) directly as part of its syntax. Metadata commands are specified via the following syntax:

$$\langle \mathit{Semgus} \rangle ::= (\mathit{metadata} \ {:}\langle \mathit{Attribute} \rangle)$$

An $\langle \mathit{Attribute} \rangle$ is either a single keyword or a keyword paired with a value appropriate for the attribute (see Section 1.8).

The mechanism for specifying metadata coincides with the definition for annotations and term attributes in the SMT-LIB Standard, except for the fact that a metadata command should not have a leading ! symbol (see Section 3.4, 3.5 of the SMT-LIB Standard, Version 2.6).

1.3 Term Type Declarations

Term-type declarations define a universe of possible terms for use in the synthesis problem (that may be further restricted by a separate grammar later on). For those coming from SyGuS, it is convenient to think of the term-type declaration a declaring a *background theory of terms* that one may later choose to further restrict when solving a particular synthesis problem, similar to how SyGuS problems allow one to restrict the considered terms within a background theory such as LIA.

Term-types are declared via the following syntax:

$$(\mathit{declare\text{-}term\text{-}types} \ (\langle \mathit{SortDecl} \rangle^{n+1}) \ (\langle \mathit{DTDecl} \rangle^{n+1}))$$

This syntax (including the definitions for subterms $\langle \mathit{SortDecl} \rangle$ and $\langle \mathit{DTDecl} \rangle$) is identical to the datatype declaration command *declare-datatypes* within SMT-LIB and SyGuS (see Section 2.9 of the SyGuS language standard, Version 2.1, and Section 4.2.3 of the SMT-LIB Standard) except for the fact that the heading command is named *declare-term-types*.

1.4 Semantic Declarations

SemGuS requires one to explicitly state the semantics for terms within the considered universe defined in §1.3; combined with the term-type declarations, one may consider these as the (custom-defined) background theory to operate over.

Semantics for terms defined in §1.3 is done using the *define-funs-rec* command, as following:

$$(\textit{define-funs-rec}\ (\langle \textit{Function\_Dec} \rangle^{n+1})\ (\langle \textit{Term} \rangle^{n+1}))$$

This syntax (including the definitions for subterms $\langle Function\_Dec \rangle$ and $\langle Term \rangle$) is identical to the command for declaring mutually recursive functions in the SMT-LIB standard (see Section 4.2.3 of the SMT-LIB standard, Version 2.6).

Although the syntax for specifying semantics in SemGuS files is permissive, SemGuS requires that semantics are defined using semantic relations and CHCs. In addition, SemGuS requires that these semantics be defined on an production-by-production basis—thus in practice, the following restrictions should be enforced by a SemGuS solver:

1. The return types of all functions declared in *declare-funs-rec* should i) contain an argument t of a term-type defined in §1.3, and ii) return a value of type *Bool*.

2. The body part $\langle Term \rangle$ should be a *match* statement that matches t over the possible constructs defined in §1.3.

In addition, SemGuS allows multiple $\langle \textit{Term} \rangle$s to be associated with a single $\langle \textit{Pattern} \rangle$ in pattern matching. This allows one to equip multiple CHCs (for cases like while loops, or nondeterminstic programs) to a single constructor. Thus the syntax for $\langle \textit{match-case} \rangle$ (taken from the SMT-LIB standard, used for defining match statements) is changed to:

$$\langle \textit{match-case} \rangle ::= (\langle \textit{Pattern} \rangle\ \langle \textit{Term} \rangle^{+})$$

All other subexpressions are identical to the SMT-LIB standard.

1.5 Synth-Fun

The *synth-fun* command declares a single actual function to synthesize. The syntax is identical to the synth-fun command in the SyGuS standard; refer to Section 2.9 of the Sygus format, Version 2.1 for details.

To synthesize multiple functions, one should have multiple *synth-fun* commands in the file, one for each function to synthesize.

1.6 Constraints

The *constraint* command declares the behavioral specification for the functions to synthesize defined using *synth-fun*. The syntax once again borrows from the SyGuS standard; refer to Section 2.9 of the Sygus standard, Version 2.1 for details.

1.7 Other Commands

SemGuS accepts other standard SMT-LIB and SyGuS commands such as variable or sort declarations. All accepted other commands coincide with the syntax in SyGuS and SMT-LIB; a full list of other accepted commands can be found in §1.9.

To keep in sync with updates to SyGuS, we omit a hardcoded syntax definition for most constructs in this part, and instead refer the reader to the corresponding sections within the SyGuS format specification.

1.8 Subexpressions

Most subexpression definitions used in SemGuS (such as $\langle Term \rangle$s, $\langle Attribute \rangle$s, etc.) are identical to their definitions in SyGuS and SMT-LIB. We give a quick overview here.

1.8.1 Literals

Literals are special sequences of characters that are mostly used to denote values and 0-ary symbols of a background theory. The definition for literals in SemGuS is identical to that in SyGuS; for further information, consult Section 2.2 of the SyGuS standard, Version 2.1, or Section 3.1 of the SMT-LIB standard, Version 2.6.

1.8.2 Symbols and Identifiers

A symbol is a non-empty sequence of alphabets, digits, and certain special characters, that may not begin with a digit and is not a reserved word. An identifier is an extension of symbols to symbols that are indexed by integer constants or other symbols. The syntax of symbols and identifiers in SemGuS is identical to SyGuS and SMT-LIB; we refer the reader to Section 2.3 and 2.4 of the SyGuS standard, Version 2.1, for further information.

1.8.3 Attributes

An attribute is either a keyword $\langle Keyword \rangle$ or a keyword with an associated value. Attributes are used to annotate terms, as well as provide metadata in the metadata command. The syntax for keywords and attributes is identical to that in SyGuS; we refer the reader to Section 2.5 of the SyGuS standard for further information. An $\langle AttributeValue \rangle$ depends on the $\langle Keyword \rangle$ it is associated with. It is up to the solver to support combinations of keywords and attribute values; the SemGuS standard does not provide a pre-defined list of keywords that must be supported.

1.8.4 Terms

Terms $\langle Term\rangle$ are used to specify grammars, constraints, semantics, and many other things. They use the same syntax as in SyGuS; we refer the reader to Section 2.7 of the SyGuS standard for further information.

Note that any term may be annotated with an attribute via the ! $\langle \mathit{Term}\rangle$ $\langle \mathit{Attribute}\rangle^{+}$ syntax.

1.9 Command Syntax

We now give an incomplete list of commands that make up the SemGuS format, focusing on a basic list of commands that we expect to be central to specifying a SemGuS problem. In addition to these commands, SemGuS supports all commands that SyGuS supports with the provision that semantics of some of these commands may be different; in essence, SemGuS diverges only from the SyGuS format through the addition of the *declare-term-types* command, with some additional semantic restrictions.

Some of the productions listed here contain nonterminals (e.g., $\langle Function\_Dec\rangle$) that are not defined in this document; for the concrete definition of these productions, we refer the reader to Section 2.9 of the Sygus specification, Version 2.1.

$$
\begin{array}{lcl}
\langle \mathit{Cmd}\rangle & ::= & (\mathit{check\text{-}synth}) \\
 & | & (\mathit{constraint}\ \langle Term\rangle) \\
 & | & (\mathit{declare\text{-}term\text{-}types}\ (\langle \mathit{SortDecl}\rangle^{n+1})\ (\langle \mathit{DTDecl}\rangle^{n+1})) \\
 & | & (\mathit{synth\text{-}fun}\ \langle \mathit{Symbol}\rangle\ (\langle \mathit{SortedVar}\rangle^{*})\ \langle \mathit{Sort}\rangle\ \langle \mathit{GrammarDef}\rangle^{?}) \\
 & | & \langle \mathit{SmtCmd}\rangle \\
 & & \\
\langle \mathit{SmtCmd}\rangle & ::= & (\mathit{declare\text{-}var}\ \langle \mathit{Symbol}\rangle\ \langle \mathit{Sort}\rangle) \\
 & | & (\mathit{declare\text{-}datatype}\ \langle \mathit{Symbol}\rangle\ \langle \mathit{Sort}\rangle) \\
 & | & (\mathit{declare\text{-}datatypes}\ (\langle \mathit{SortDecl}\rangle^{n+1})\ (\langle \mathit{DTDecl}\rangle^{n+1})) \\
 & | & (\mathit{declare\text{-}sort}\ \langle \mathit{Symbol}\rangle\ \langle Numeral\rangle) \\
 & | & (\mathit{define\text{-}fun}\ \langle \mathit{Symbol}\rangle\ (\langle \mathit{SortedVar}\rangle^{*})\ \langle \mathit{Sort}\rangle\ \langle Term\rangle) \\
 & | & (\mathit{define\text{-}sort}\ \langle \mathit{Symbol}\rangle\ \langle \mathit{Sort}\rangle) \\
 & | & (\mathit{define\text{-}funs\text{-}rec}\ (\langle Function\_Dec\rangle^{n+1})\ (\langle Term\rangle^{n+1})) \\
 & | & (\mathit{set\text{-}info}\ \langle Keyword\rangle\ \langle \mathit{Literal}\rangle) \\
 & | & (\mathit{set\text{-}logic}\ \langle \mathit{Symbol}\rangle) \\
 & | & (\mathit{set\text{-}option}\ \langle Keyword\rangle\ \langle \mathit{Literal}\rangle)
\end{array}
$$