The Graph State File Specification [1]

By

Kalpana Bharadwaj

revised by

Dick Stevens

January 27, 2000, May 26, 2001

The Processing Graph Method Tool (PGMT) product is being released under the GNU General Public License Version 2, June 1991 and related documentation under the GNU Free Documentation License Version 1.1, March 2000.
http://www.gnu.org/licenses/gpl.html

A GSF is an ASCII file which describes the layout, and associated data values, of graph objects that comprise a PGM2 application. This description is sufficient to store and retrieve a complete graph across different invocations of the graph editor application. This document is a formal specification of the GSF language. The following is a description of the structure of this document:

1. Notation and Vocabulary.
2. Language Design - An abstract.
3. GSF in the EBNF notation.
4. GSF in the BNF notation.
5. Appendix A - User-defined Prototypes.
6. Appendix B - Predefined Prototypes.
7. Appendix C - Instances of Prototypes.
8. Appendix D - Example of a gsf.
9. Appendix E - A Syntax Checker for the gsf.

Notation and Vocabulary

Our grammar has four components:

- A set of tokens, known as *terminal* symbols.
- A set of *nonterminals*.
- A set of *productions* where each production consists of a nonterminal, called the *left side* of the production, a symbol "::=", and a sequence of tokens and/or nonterminals, called the *right side* of the production.
- A designation of one of the symbols as the start symbol.

Notation:

- Terminal symbols are in typewriter fonts.
- **Nonterminals** are in bold.
- *Undefined lexical classes* are in *italics*.
- [ ] are meta-symbols which denote choice or repetition of syntactic classes.
- Suffix * represents 0 or more repetitions of the preceding construct.
- Suffix $^+$ represents 1 or more repetitions of the preceding construct.

- Suffix ? represents 0 or 1 repetitions of the preceding construct.
- _ denotes choice.

GSF in the EBNF notation

Pgsf :: = capsule { **Banner Exterior** [**NodeProto** | **Instance** | **Arc**]*}

Banner :: = banner {filename { : *string* : } author { : *string* : }

revision { : *string* : } purpose { : *string* : } graphtype *number* ; }

Exterior :: = exterior { **Prototype** [**PortAssoc**]$^?$ [**InclForm**]$^?$

[**Function**]$^?$ [**TypeForm**]$^?$ }

TypeForm :: = type { [**Name** @ **PathName** ;]$^+$ }

PortAssoc :: = [**InAssoc**]$^?$ [**OutAssoc**]$^?$ | [**OutAssoc**]$^?$ [**InAssoc**]$^?$

InAssoc :: = input { [**Name** <=> *label* . **Name** ;]$^+$ }

OutAssoc :: = output { [**Name** <=> *label* . **Name** ;]$^+$ }

Function :: = function { [**BaseType** **Name**

< [**BaseType** [, **BaseType**]*]$^?$ > ;]$^+$ }

InclForm :: = inclgraph { [**Name** @ **PathName** ;]$^+$ }

NodeProto :: = [transition | gvar | queue] { **Prototype**}

Prototype :: = prototype **ProtName FamName** ; [gips { **Gips** }]$^?$

[**Inports**]$^?$ [**Outports**]$^?$ [**TrStmt**]$^?$

Gips :: = [**Name** : **Mode** ;]$^+$

Mode :: = < **TokHt** , **BaseType** >

TrStmt :: = trstmt { : *string* : }

Instance :: = *label*[[**Range**]]* location < **Location** > =

[transition | place | inclgraph] **ProtCall FamInit**

[{ [gips {**Bindings**}]$^?$ [initval {**Value**}]$^?$ }]$^?$;

Bindings :: = [**Name** = **Value** ;]*

ProtName :: = **Name** [< **Template** [, **Template**]* >]$^{?}$

ProtCall :: = **Name** [< **BaseType** [, **BaseType**]* >]$^{?}$

FamName :: = [fmly (**FormalArg** [, **FormalArg**]*)]$^{?}$

FamInit :: = [fmly (< **Mode** > [, **Mode**]*)]$^{?}$

Arc :: = arc[[**Range**]]* [bends [< **Points** >]$^{+}$]$^{?}$

{ **Connect** - > **Connect** [when **Expr**]$^{?}$ }

Connect :: = *label*[[**Expr**]]*.**Name**[[**Expr**]]*

Range :: = [**Expr** : **Name** : **Expr**]

Inports :: = inport { [**Name** **Mode** [[**Range**]]*

category **Cat** ;]$^{+}$ }

Outports :: = outport { [**Name** **Mode** [[**Range**]]*

category **Cat** ;]$^{+}$ }

Cat :: = place | transition

Value :: = [[**Range**]]* leaf [**Expr**] | **NestedString**

BaseType :: = **Name**

NestedString :: = { [**Expr** [**Expr**]*]$^{?}$ } |

{ [**NestedString** [**NestedString**]*]$^{?}$ }

TokHt :: = **Expr**

Location :: = *number* , *number*

Points :: = *number* , *number*

Expr :: = *target language specific implementation*

Name :: = *defined by the implementation*

PathName :: = " **Name** "

Template :: = **Name**

FormalArg :: = < **Name** , **Name** >

GSF in the BNF notation

Pgsf :: = capsule { **Banner Exterior Specs** }

Specs :: = *null* | **Specs NodeProto** | **Specs Instance** | **Specs Arc**

Banner :: = banner { filename *string* author *string* revision *string*
purpose *string* graphtype *number* ; }

Exterior :: = exterior { **Prototype Pass Pincl Pfun Ptype** }

Pass :: = *null* | **Pass PortAssoc**

Pfun :: = *null* | function { **Func** }

Ptype :: = *null* | **TypeForm**

Pincl :: = *null* | **InclForm**

TypeForm :: = type { **BasePrim** ; }

PortAssoc :: = input { **Assoc** } | output { **Assoc** }

Assoc :: = **Name** <=> *label* . **Name** ; |
Assoc Name <=> *label* . **Name** ;

Func :: = **BaseType Name** < **BasePrim** > ; |
Func BaseType Name < **BasePrim** > ;

BasePrim :: = *null* | **BaseType BaseAux**

BaseAux :: = *null* | **BaseAux** , **BaseType**

InclForm :: = inclgraph { **InclIden** }

InclIden :: = **Name @PathName** ; | **InclIden Name @PathName** ;

NodeProto :: = transition { **Prototype** } | gvar { **Prototype** } |
queue { **Prototype** }

Prototype :: = prototype **ClassName FamName** ;
Gos INP OUTP TrStmt

Gos :: = *null* | gips { **Gips** }

Gips :: = **Name** : **Mode** ; | **Gips Name** : **Mode** ;

INP :: = *null* | inport { **Port** }

OUTP :: = *null* | outport { **Port** }

Port :: = **Name Mode RangeSeq Cat** ; | **Port Name Mode RangeSeq Cat** ;

RangeSeq :: = *null* | **RangeSeq** [**Range**]

Range :: = **Expr** <= **Name** <= **Expr**

Cat :: = category transition | category place

Mode :: = < **Expr** , **BaseType** >

TrStmt :: = trstmt *string*

Location :: = *number* , *number*

Points :: = *number* , *number*

Instance :: = *label* **RangeSeq** location < **Location** > = **Kind ClassCall FamInit Params**;

ClassCall :: = **Name TypeSeq**

ModeSeq :: = *null* | < **Mode ModeAux** >

ModeAux :: = *null* | **ModeAux** , **Mode**

Kind :: = transition | place | incgraph

Params :: = *null* | { **Gparam Iparam** }

Gparam :: = *null* | gips { **Bindings** }

Iparam :: = *null* | initval { **Value** }

Value :: = **RangeSeq** leaf [**Expr**] | **NestedString**

Bindings :: = **Name** = **Value** ; | **Bindings Name** = **Value** ;

NestedString :: = { **NestPrim** }

NestPrim :: = *null* | **NestPrim NestAux**

NestAux :: = **Expr** | { **NestPrim** }

ClassName	:: =	**Name TempPrim**
TempPrim	:: =	*null* | < **Template TempAux** >
TempAux	:: =	*null* | **TempAux** , **Template**
BaseType	:: =	**Name TypeSeq**
TypeSeq	:: =	*null* | < **BaseType BaseAux** >
Arc	:: =	arc **RangeSeq BendSeq** { **Link** }
BendSeq	:: =	*null* | bends **PtSeq**
PtSeq	:: =	< **Points** > | **PtSeq** < **Points** >
Link	:: =	**Connect** -> **Connect** | **Connect**-> **Connect** when **Expr**
Connect	:: =	*label* **ExSeq** . **Name ExSeq**
ExSeq	:: =	*null* | **ExSeq** [**Expr**]
FamName	:: =	*null* | fmly (**ForArg ForArgAux**)
ForArgAux	:: =	*null* | **ForArgAux** , **ForArg**
ForArg	:: =	< **Name** , **Name** >
FamInit	:: =	*null* | fmly (**Mode ModeAux**)

Footnotes:

[1] I thank Rudolph Krutar and David Kaplan for reviewing the grammar.