In- and output programming interface

Jelle Hellings

June 15, 2011

1 Introduction

This document describes the in- and output interfaces provided by these programs.

2 Program interfaces

Almost all programs developed for this project read input from some file and write the results to another file. For this input and output we have used various binary data formats. We shall describe these formats in this section. Every used format starts with a unique format identifier. The following identifiers are currently in use: `tree`, `dag`, `dagfp`, and `dagfps` (see `exbisim/header/common/format.hpp`). The `dagfp` format provides a $\mathcal{L}$ representation of graphs; the `dagfps` format provides a $\mathcal{L}_S$ representation of graphs.

In the current implementation all numbers (represented by $\mathbb{N}$ in the format grammars) are limited to 32-bit unsigned integers. This limits the possible values to the range $[0, 2^{32} - 1)$; this also limits the number of nodes in a tree or in a graph. All integers are stored in machine-dependent order (little endian for the common Intel x86 architecture).

2.1 The format `tree`

The `tree` format is described by the following grammar:

Tree ⟵ `tree` Node
Node ⟵ `o` Label Node* `c`
Label ⟵ $\mathbb{N}$

Trees in `tree` format start with the format identifier `tree`. The content of a tree consists of a single root node. A node consists of an open tag `o` and close tag `c`; a node has a single label. Node open `o` and close `o` tags are represented by a byte in input; with ASCII values 'o' (111) and 'c' (99) (see `exbisim/header/common/treetag.hpp`).

2.2 The format `dag`

The `dag` format is described by the following grammar:

$$\begin{aligned} \text{Dag} &\longleftarrow \texttt{dag}\ \text{NumNodes}\ \text{Node}^* \\ \text{Node} &\longleftarrow \text{Label}\ \text{NumChildren}\ \text{ChildNodeIdentifier}^* \\ \text{NumNodes} &\longleftarrow \mathbb{N} \\ \text{Label} &\longleftarrow \mathbb{N} \\ \text{NumChildren} &\longleftarrow \mathbb{N} \\ \text{ChildNodeIdentifier} &\longleftarrow \mathbb{N} \end{aligned}$$

Directed acyclic graphs in `dag` format start with the format identifier `dag`. The content of a directed acyclic graph consists of the number of nodes n followed by exactly n nodes. Each node consists of a label, the number of child nodes c followed by exactly c child node identifiers. The following interpretation is placed on this structure:

1. The i-th node in the list has identifier $i - 1$ (thus we have node identifiers in the range $[0, \text{NumNodes})$).
2. All children of node n with identifier i must have a smaller node identifier then node n; this assures that the directed acyclic graph is reverse-topological ordered.

2.3 The format `dagfp`

The `dagfp` format provides a $\mathcal{L}$ representation of graphs. The `dagfp` format is described by the following grammar:

$$\begin{aligned} \text{DagFP} &\longleftarrow \texttt{dagfp}\ \text{NumNodes}\ \text{Node}^* \\ \text{Node} &\longleftarrow \text{Label}\ \text{NumParents}\ \text{ParentNodeIdentifier}^* \\ \text{NumNodes} &\longleftarrow \mathbb{N} \\ \text{Label} &\longleftarrow \mathbb{N} \\ \text{NumParents} &\longleftarrow \mathbb{N} \\ \text{ParentNodeIdentifier} &\longleftarrow \mathbb{N} \end{aligned}$$

Directed acyclic graphs in `dagfp` format are similar to directed acyclic graphs in `dag` format. The main difference is that in directed acyclic graphs in `dagfp` format each node has an edge adjacency list pointing to all parent nodes instead of an edge adjacency lists pointing to all child nodes.

Directed acyclic graphs in `dagfp` format start with the format identifier `dagfp`. The content of such a directed acyclic graph consists of the number of nodes n followed by exactly n nodes. Each node consists of a label, the number of parent nodes p followed by exactly p parent node identifiers. The following interpretation is placed on this structure:

1. The i-th node in the list has identifier $i - 1$ (thus we have node identifiers in the range $[0, \text{NumNodes})$).
2. All parents of node n with identifier i must have larger node identifiers then node n; this assures that the directed acyclic graph is reverse-topological ordered.

2.4 The format `dagfps`

The `dagfps` format provides a $\mathcal{L}_S$ representation of graphs. The `dagfps` format is described by the following grammar:

$$
\begin{aligned}
\text{DagFPS} &\longleftarrow \texttt{dagfps}\ \text{NumPartitionBlocks}\ \text{PartitionBlock}^* \\
\text{PartitionBlock} &\longleftarrow \text{NumNodes}\ \text{Rank}\ \text{Label}\ \text{Node}^* \\
\text{Node} &\longleftarrow \text{NumParents}\ \text{ParentNodeIdentifier}^* \\
\text{NumPartitionBlocks} &\longleftarrow \mathbb{N} \\
\text{NumNodes} &\longleftarrow \mathbb{N} \\
\text{Rank} &\longleftarrow \mathbb{N} \\
\text{Label} &\longleftarrow \mathbb{N} \\
\text{NumParents} &\longleftarrow \mathbb{N} \\
\text{ParentNodeIdentifier} &\longleftarrow \mathbb{N}
\end{aligned}
$$

Directed acyclic graphs in `dagfps` format are similar to directed acyclic graphs in `dag` or `dagfp` format. The main difference with the other directed acyclic graph formats is that the `dagfps` format places more constraints on the ordering of nodes: nodes are grouped in partition blocks wherein every node has *at least* the same rank and label. These partition blocks of rank r and label l are lexicographically ordered on (r, l).

Directed acyclic graphs in `dagfps` format start with the format identifier `dagfps`. The content of such a directed acyclic graph consists of the number of partition blocks b followed by exactly b partition blocks. Each partition blocks consists of a number of nodes n placed in the group, a rank and a label; followed by exactly n nodes. Each node consist of a number of parents p followed by exactly p parent node identifiers. The following interpretation is placed on this structure:

1. The i-th node in the list has identifier $i - 1$.
2. All parents of node n with identifier i must have larger node identifiers then node n; this assures that the directed acyclic graph is reverse-topological ordered.
3. The rank of a node is equal to the length of the longest path to a leaf node in the directed acyclic graph.
4. The nodes in the message are sorted on increasing $(\text{rank}, \text{label})$.