Title:
Lost in Translation: Evaluating the Robustness of a Data Format and its Read/Write Machinery

Authors:
John K. Johnstone, jkj@uab.edu, Computer Science and Ophthalmology, UAB

Keywords:
CAD dataset, Data management, Data format, Reader, Writer, Verification, Optic nerve head dataset.

DOI: 10.14733/cadconfP.2024.227-231

Introduction:

This abstract considers mechanisms to evaluate the correctness of a new data format for data storage and its read-write machinery, as well as to guarantee the preservation of information content when translating between two data formats. This issue has increasing importance in today's world of large datasets, where the corruption of data has large consequences for computation. Formal mechanisms for data management and provenance are also increasingly being emphasized by the NIH and NSF [2, 3], illustrating the importance of this issue. The focus is on CAD datasets such as polygon meshes and point clouds, but the mechanisms generalize to other datasets. A major purpose of the abstract is straightforward: to highlight the issue of robustness and verification of data and, in the full paper, to consider a case study of the use of these tests in the development of a new data format and its read/write machinery, and the incorporation of these tests into the software lifecycle.

These issues were highlighted for us recently in developing a new data format for optic nerve head point clouds in a biomedical context (ophthalmology), where there are high expectations for the sanctity and provenance of the underlying data and where the datasets are large and complex.

Consider the storage of data in a certain data format and its read/write machinery. Data is stored externally in a data file using a data format, and it is stored internally, in code, in a data structure. This symbiotic relationship between the internal and external representations is subtle, yet the preservation of information content in the data is obviously crucial as it is translated between internal and external representations, and between different data formats. There are many moving parts that must be coordinated here: the data format, the internal data structure, the reader, the writer, and any translators between different data formats. This issue becomes most tangible when developing and introducing a new data format: can the data structure be restored from the new data format? will the information content of the data stored in previous formats be preserved when translated to the new format?

The sanctity of data, and the importance of preserving its information content as it is massaged into different forms, demands that this issue be treated more deliberately and more formally. We introduce several tests that can be applied to guarantee that no information is lost in translating between internal and external representations of data, and between different external representations of data.

As a concrete example, consider the triangle mesh as the dataset of interest. There are many internal representations of a triangle mesh (e.g., halfedge, winged edge, list of vertices/faces) and there are many external representations of a triangle mesh (e.g., PLY, OBJ, OFF formats [7, 8, 9]). Choose one of the

internal representations of a triangle mesh (say, halfedge) and let I be the universe of valid internal half-edge representations of all triangle meshes. Choose one of the external representations of a triangle mesh (say, PLY) and let E be the universe of valid external PLY representations of all triangle meshes. A (PLY, halfedge) writer $w : I \to E$ is code whose input is a halfedge internal representation of a triangle mesh and whose output is a PLY external representation for a triangle mesh. A (PLY, halfedge) reader $r : E \to I$ is code whose input is a PLY external representation file for a triangle mesh and whose output is a halfedge internal representation of a triangle mesh. Note that there are three hyper-parameters to a reader/writer: the structure of interest (in this case, triangle mesh), the external representation of this structure (PLY), and the internal representation of this structure (half-edge). We are interested in the correctness of the translation from internal to external representation and back.

We are also interested in the correctness of the translation between external representations in two different data formats. This adds a fourth hyper-parameter: the second external representation of this structure (e.g., OBJ). Choose a second external representation of a triangle mesh (OBJ) and let E_2 be the universe of valid external OBJ representations of all triangle meshes. A (PLY, OBJ) translator $t : E \to E_2$ is code whose input is a PLY file for a triangle mesh and whose output is an OBJ file for the same triangle mesh. A (OBJ, PLY) translator $t_2 : E_2 \to E$ is code whose input is an OBJ file for a triangle mesh and whose output is a PLY file for the same triangle mesh. We also want to test the integrity of these translators.

The rest of the abstract is structured as follows. We first discuss internal tests, which focus on the preservation of data in its internal form. We then discuss external tests, which focus on the preservation of data in its external form. We conclude with translation tests, which focus on data preservation while translating between two different data formats.

Internal test:

An internal test is a fundamental test of a data format and its read/write machinery, since it reflects a fundamental purpose of a data file: as a temporary repository for the internal data structure. An external data file allows a computation to be restarted after a temporal break; an external data file also allows the underlying structure to be communicated to another processor so that it can continue the computation. So the file format records a snapshot of the meaningful elements of the data structure, allowing i at time/place t/p to be restored as i at time/place t+delta/p or time/place t+delta/q. The basic question that must be answered in the affirmative is: if you store a data structure x for a while in a data file f, can you restore x from f?

Definition 1
Consider a dataset of interest (e.g., a polygon mesh or a point cloud).
Fix an internal data structure for this dataset (e.g., half-edge representation of a mesh).
Fix a data format for this dataset (e.g., OBJ representation of a mesh).
Let w be a writer, which translates an internal data structure to a data file in the data format.
Let r be a reader, which translates a data file in the data format to an internal data structure.
The data format's IO machinery passes the ***internal test*** *on internal data x if*

$$r(w(x)) = x$$

The internal test works on an object x in the internal data structure. There are at least three sources for x. Most purely, x may be built directly in the code, using some algorithm to build x. For example, using a graph-cut algorithm [5] to segment an image to generate a point cloud x, if the dataset of interest is a point cloud; or using Poisson reconstruction [1] of a point cloud to generate a triangle mesh x, if the dataset of interest is a triangle mesh; or using a cubic fitting algorithm [4] to generate a B-spline curve from a point cloud, if the dataset of interest is a B-spline curve.

x may also be built randomly with the correct structure. This is a good way to stress-test the reader/writer/format/structure quartet, since randomness helps to expose weakness.

Another simple way to build x is by reading a file f: $x = r(f)$.

Finally, note that there is an equality test at the heart of the internal test. However, when the components of the internal data are floating point, only approximate equality is expected, given finite precision.

External test:

An internal test is possible after creating an instance of the internal data structure. Therefore, it is natural to apply as one writes the structure out to a data file, to guarantee that the write is robust. But if you instead start with a data file, a different test may be more natural. The external test is symmetric to the internal test, starting from the external data file rather than from the internal data structure. It tests whether you can read a data file and then later restore the same data file by writing. Since whitespace differences in plain-text data files are irrelevant, the external test ignores these differences.

Definition 2
Consider a dataset of interest.
Fix an internal data structure for this dataset.
Fix a data format for this dataset.
Let w be a writer, which translates an internal data structure to a data file in the data format.
Let r be a reader, which translates a data file in the data format to an internal data structure.
Let squish be a function that strips extra whitespace from each line of a file.
*The data format's IO machinery passes the **external test** on binary data file f if*

$$w(r(f)) = f$$

*The data format's IO machinery passes the **external test** on plain-text data file f if*

$$squish(w(r(f))) = squish(f)$$

Translation test:

Internal and external tests measure the robustness of the read/write machinery of a single format. Translation tests measure the robustness of a new format, by measuring its robustness in preserving the information in an existing format.

It might seem that the robustness of the translation between two formats is implied by the robustness of the read/write machinery of each format. After all, if you can read format 1 robustly into the internal data structure, and write format 2 robustly from the internal data structure, this seems to imply that you can translate from format 1 to format 2, using this reader/writer pair. Similarly the dual reader/writer pair (reading format 2 and writing format 1) would seem to establish the robustness of translation from format 2 to format 1.

The subtlety is that one format may contain more information than another, and this is often the case. For example, both PLY [6, 7] and OBJ formats [8] for polygon meshes offer the opportunity for much more information to be stored about a mesh than the OFF format [9]. The OFF format focuses on edges, faces, and optionally edges, while the OBJ and PLY formats also allow colour, texture, and other information to be encoded. This is also the case with the new format for optic nerve head point clouds that we are developing: it records only a subset of the data in an earlier format, because some of this original data has become vestigial and is no longer used in morphometric algorithms.

This poses a different robustness issue than the original reader/writer pair robustness measured by internal and external tests. Now one is interested in whether the subset of common information is preserved. In the PLY/OFF case, this would be the preservation of vertex/face/edge information.

Definition 3
Let $\mathcal{F}$ and $\mathcal{G}$ be two data formats for encoding the same dataset $\mathcal{X}$.[1]
Let X be the universe of datasets that are representable in both formats.[2]
Let F_1 be the universe of valid representations of X in the first format $\mathcal{F}$.
Let F_2 be the universe of valid representations of X in the second format $\mathcal{G}$.
Let $toy : F_1 \to F_2$ be a translator of a data file in format $\mathcal{F}$ to the equivalent data file in format $\mathcal{G}$.
Let $tox : F_2 \to F_1$ be a translator from format $\mathcal{G}$ to $\mathcal{F}$.
Let squish be a function that strips extra whitespace from each line of a file.
The data formats' translation machinery passes the ***translation test*** *on the pair of files f_1/f_2 of the same dataset if:*

$$squish(tox(toy(f_1))) = squish(f_1)$$

and

$$squish(toy(tox(f_2)) = squish(f_2)$$

This translation test should be applied as each dataset is translated from one format to another. We note that consideration of the translation issue, as formalized in translation tests, also pushes the designer of a data format to consider the coverage of a data format more thoughtfully, since the gaps become more visible in identifying the universe of datasets representable both in the new format and earlier formats.

Case study:
The next issue to consider is the implementation of the internal/external/translation tests and the design of readers, writers, translators, and data formats that are conducive to passing these tests. This will be addressed in the full paper. As a case study, the internal, external, and translation tests have been implemented for a structure in optic nerve head morphometry: the ONH dataset, using a new data format developed recently. The full paper will include more details on this case study, and its insights into the implementation of the verification tests.

Conclusions:
The main contribution of this abstract is a deliberate focus on the issue of robust testing of reader/writer pairs for a data format, and on robust testing of translators between data formats. Once this focus is applied deliberately and carefully, the path to these tests is natural, and can be applied universally in these environments.

We suggest adding the option to incorporate internal/external tests during the reading/writing of a dataset, as a conservative step to guarantee that no information is lost. It is also natural to add translation tests during translation of datasets between formats.

Future work includes adoption and application of these tests in a wider spectrum of applications. In particular, we are interested in parametric curve datasets and formats, where there is little work.

References:
[1] Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson Surface Reconstruction. Eurographics Symposium on Geometry Processing 2006.

[1] For example, $\mathcal{F}$ and $\mathcal{G}$ might be PLY and OFF formats for storing a mesh.
[2] For example, X might be polygon meshes that only contain pure geometric information about the vertex/edge/face of the mesh, without extra information such as texture coordinates or colours, so that all the information about the mesh in the PLY file is also representable in OFF format.

[2] 2023 NIH Data Management and Sharing Policy:
https://grants.nih.gov/grants/guide/notice-files/NOT-OD-21-013.html

[3] NSF data management and sharing requirements:
https://new.nsf.gov/policies/pappg/23-1/ch-11-other-post-award-requirements#11D4

[4] Piegl, L., Tiller, W.: The NURBS Book. Springer, 1997. http://dx.doi.org/10.1007/978-3-642-59223-2

[5] Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22:8, 2000, 888-905. http://dx.doi.org/10.1109/34.868688

[6] The Stanford 3D Scanning Repository: https://graphics.stanford.edu/data/3Dscanrep

[7] Turk, G.: The PLY Polygon File Format.
Archived at https://gamma.cs.unc.edu/POWERPLANT/papers/ply.pdf.

[8] Wavefront OBJ format: https://en.wikipedia.org/wiki/Wavefront_.obj_file

[9] OFF format. https://en.wikipedia.org/wiki/OFF_(file_format)