

Toward a Generic Subsetting Engine

Robert O. Morris

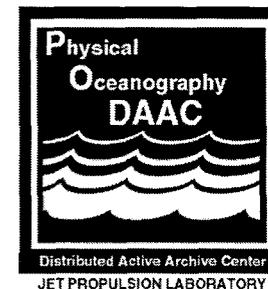
Jet Propulsion Laboratory

California Institute of Technology

March 15, 2001

Next

Credits

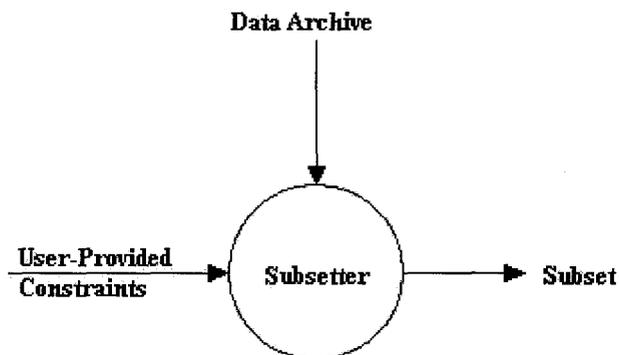


- This work was funded by EOSDIS, the Data and Information System of NASA's Earth Observing System. The work was performed at the Physical Oceanography Distributed Active Archive Center at the Jet Propulsion Laboratory.
- Copyright 2001, California Institute of Technology
- Sponsored by the US Government, NASA Contract NAS7-1260. All rights reserved.

Back

Forward

What a Subsetter Does



- Generic Constraints are:
 - Instrumental
 - Temporal
 - Spatial or
 - Parametric
- Each Generic Constraint must be explicitly provided or implied
- Dataset-Specific constraints may also be necessary
- We dictate that constraints only specify what data to include in the subset, not how it is to be processed

[Back](#)

[Forward](#)

GOALS

Develop a subsetting model that:

- Is highly Archive/DAAC independent
- Provides for output in a range of different file formats by default
- Allows for custom tailoring of the supported file formats on a DAAC-specific basis (within limits)

Back

Forward

Project Has Dual Focus

- Develop a subsetting model that works for JPL's PO.DAAC
- Sufficiently generalize basic concepts to allow model to be used at other DAAC's

[Back](#)

[Forward](#)

Method: Object-Oriented Analysis and Design

- Encapsulate dataset-specific functionality within a series of dataset-specific objects that have uniform interfaces
- Design and Apply generic rules that handle commonly occurring cases
- Allow overriding (through inheritance) of the generic rules to handle unusual cases

Back

Forward

Limitations

- We do not consider *processing* of data, only subsetting proper
 - exception is casting to types necessary for target format
- No support for point data

Handles:

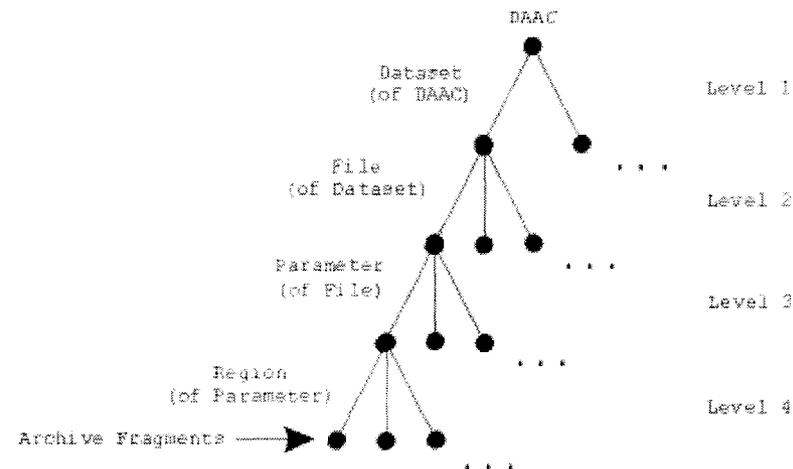
- Any data that *can be* represented as n-dimensional arrays of numbers
- Tabular data (eg, spreadsheet-like)
- Any associated metadata

Back

Forward

Archive Hierarchy

- Archives can be represented hierarchically



- Subset can be isolated via a series of 4-tuples through an archive [Dataset, File, Parameter, Region]
 - We call these *Path Descriptors* or PD's
 - This is a dataset-independent intermediate representation of a subset
 - They serve as input to a back-end that translates them to files in a target format
- The Subsetter must provide a mapping that yields PD's given the

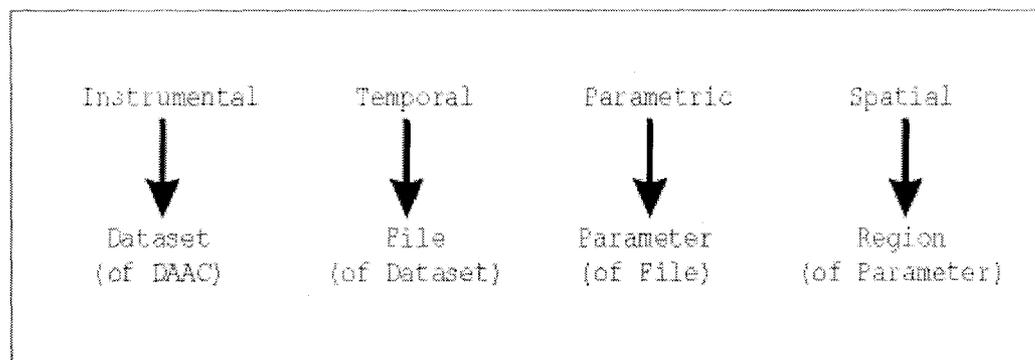
constraints

Back

Forward

Mapping from Constraints to Path Components of the Archive

- Typical example

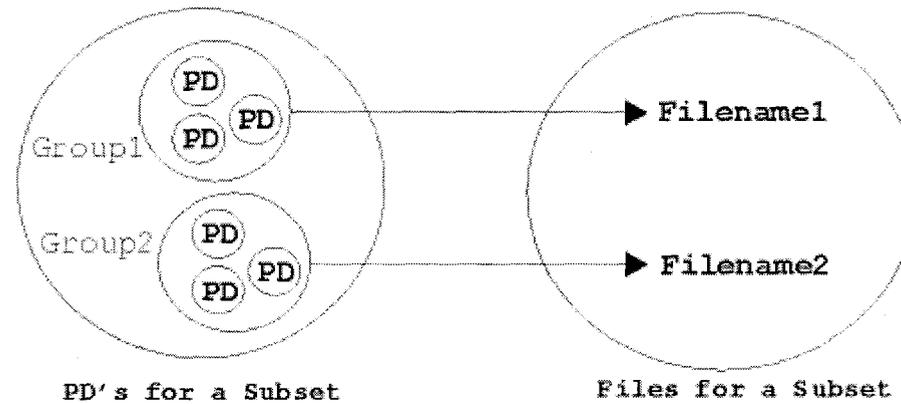


- Actual mappings are dataset-specific
 - Does **not** imply there is no code sharing between the mappings for different datasets
- Any constraint mapping that is *onto* is appropriate
- Once we have the mapping, we have the means of generating a subset represented as a series of PD's.

[Back](#)[Forward](#)

PD Grouping

- A series of PD's by themselves do not adequately describe our subset
- PD's must be grouped

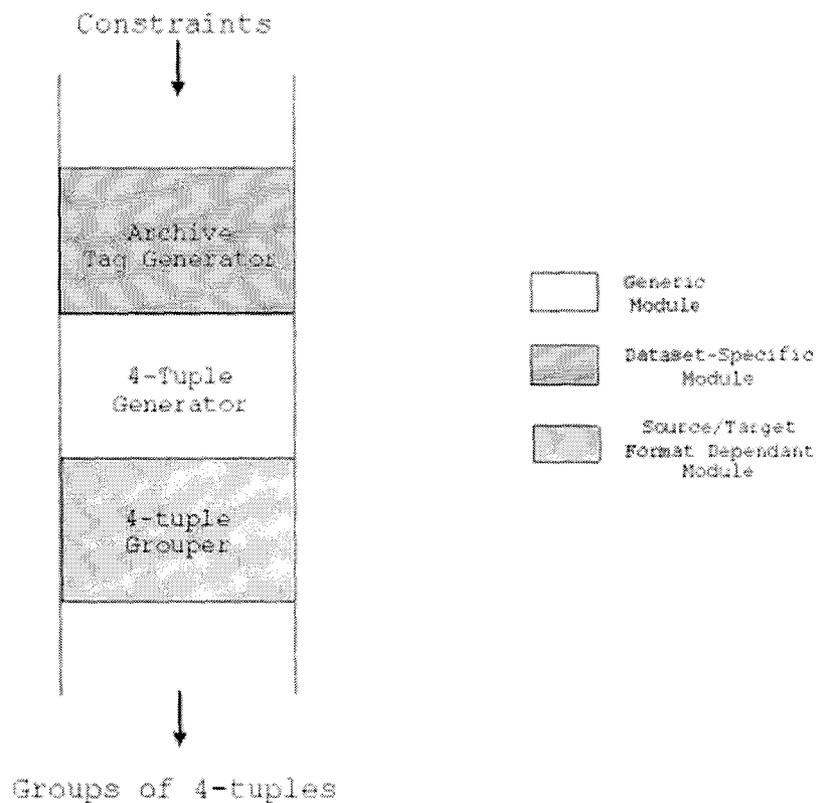


- Each group must be associated with an output filename
- PD's describe *what* goes into a subset
- Grouped PD's describe *where* the components of a subset are stored
- Given the grouped PD's, the back-end of the system determines *how* to complete the translation process

[Back](#)[Forward](#)

The Constraint Compiler

- Pipeline architecture creates the grouped Path Descriptors and associates each group with an output filename



- Archive Tag Generator isolates subset-relevant path fragments through the Archive Hierarchy Tree
 - Uses dataset-specific objects one per level in the AHT
 - Objects associated with a single level for all datasets have the same interface

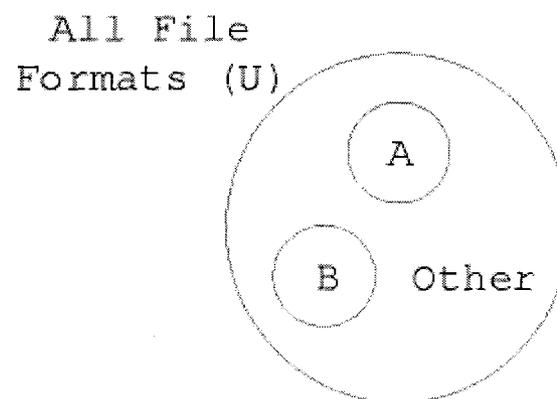
- This allows for dynamic and transparent plug-and-play
- 4-Tuple generator walks the path fragments and emits a 4-tuple each time a leaf is encountered
- Then PD's are grouped

Back

Forward

Basis for Generic Grouping/Naming Rules

- Basis formed by partitioning the Universe of File Formats according to similiar capabilities
- Construct table based upon those capabilities
- Example:



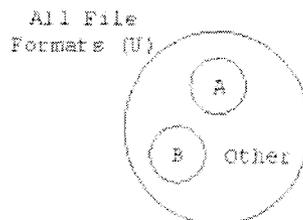
- A = File formats capable of containing any number of n-dimensional arrays and tabular data
- B = File formats capable of containing only a single 2-dimensional array
- Other = $U - A - B$

Back

Forward

Constructing Table for Generic Grouping/Naming Rules

- Example (Continued)



- Construct table based upon the partitioning

Source Format in	Target Format in	Grouping Rule	Filename Generation Rule
B	B	rasterGroup	cloneName
A	A	fileGroup	cloneName
B	A	rasterSequenceGroup	mergeName
A	B	rasterGroup	forkName
Other	Translation Disallowed (Source = Target)	fileGroup	cloneName

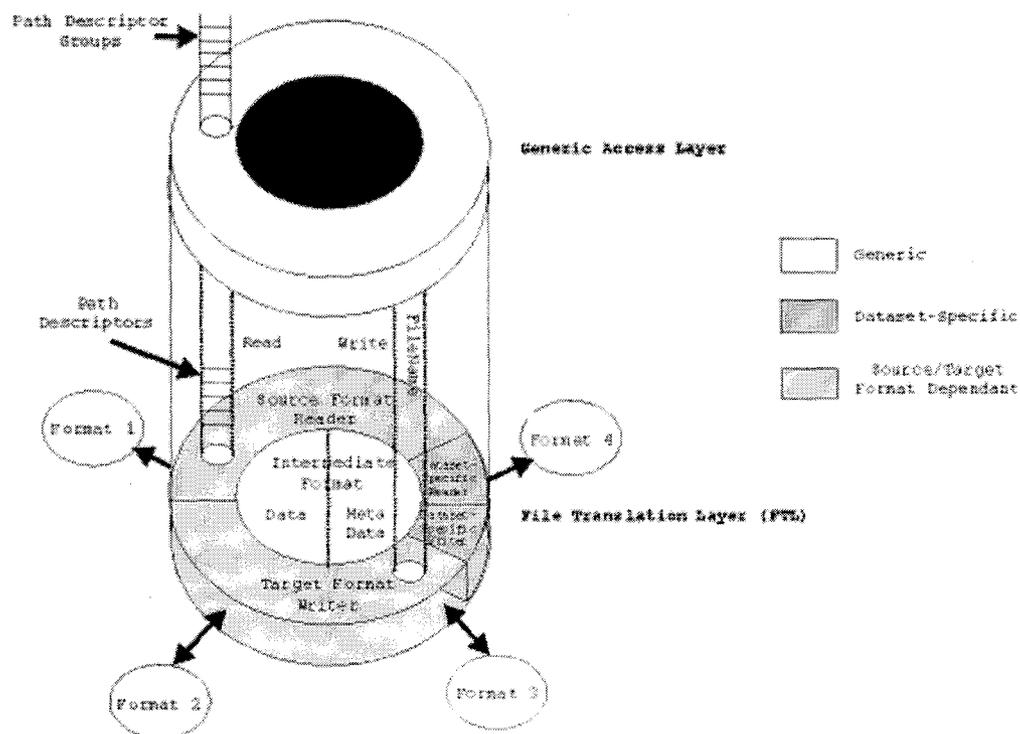
- Many other partitionings of U could form an appropriate basis for Generic Rules
- Generic rules can be overridden by constructing a DAAC-specific partitioning
- Can also be overridden in a dataset-specific manner

Back

Forward

Subset File Generator

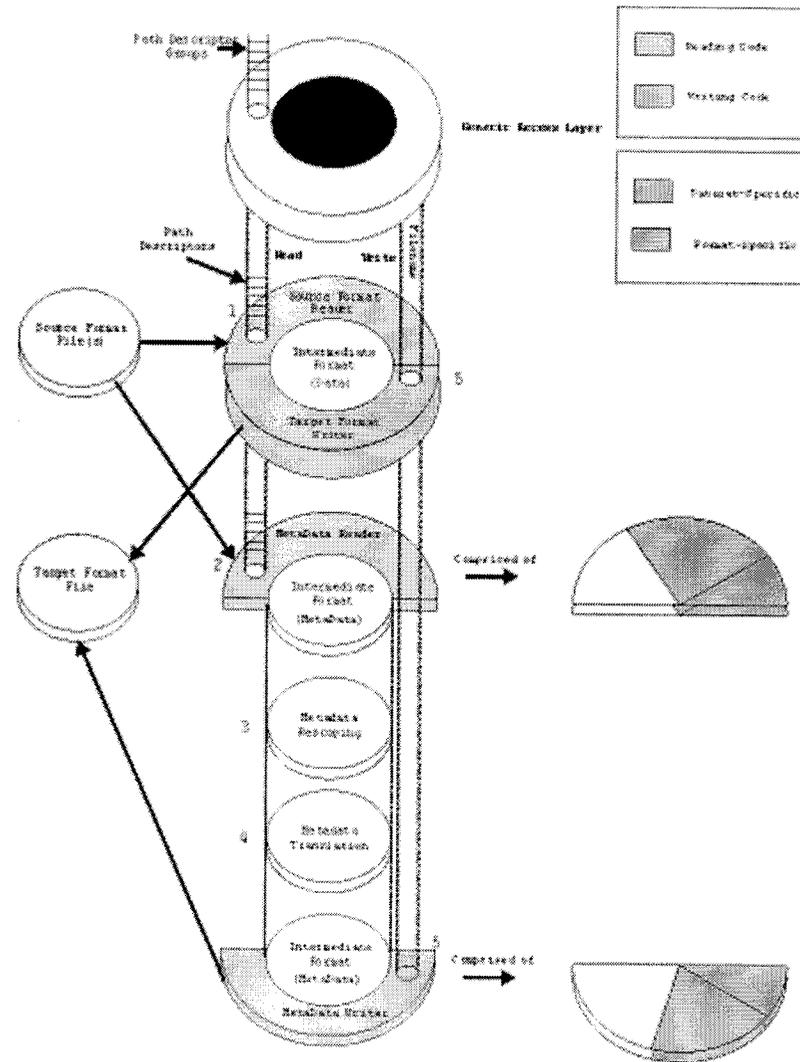
- Generic Access Layer receives a single PD Group
- Forms File Translation Layer (FTL) to handle generation of a subset file
- Then through a uniform interface to the FTL
 - Initiates read of data/metadata into an intermediate format
 - writes intermediate format out to a single file in the target format



Back

Forward

File Translation Layer Breakdown



Back

Forward

File Translation Layer Breakdown (Con't)

- Chronological order of events
 1. Given PD's, data is read into the intermediate format from one or more source files
 - format-specific reader
 2. Given PD's, metadata is read in from those same source files
 - The metadata reader is format specific *and* potentially dataset-specific
 - metadata from multiple input files will be appropriately scoped
 - Format-specific structural metadata will be encapsulated and shared
 3. Scope is shifted if necessary to accomodate target format
 - Not necessary if source and target formats come from same format set
 4. Metadata is translated in a format-specific and potentially dataset-specific basis
 5. Intermediate format used to write output files in target format

Back

Start Over