Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

An IRAF-Independent Interface for Spatial-Region Descriptors

D. Schmidt

Smithsonian Astrophysical Observatory, 60 Garden St., Cambridge, MA 02138

Abstract. Regions for spatial masking are a valued part of PROS. To move toward open analysis software, we plan to make the PROS regions subsystem available from processes outside of IRAF. We also plan to make the region descriptor parsing system available for purposes beyond the creation of masks, such as an interface to an image display program, accepting a region descriptor and generating commands to display the region on an image.

Last year, we put in place support for arbitrary actions using region descriptors. We discuss this architecture and its implications for opening up the use of regions. We then discuss the challenge of giving the subsystem an IRAF-independent interface.

1. Introduction

Limiting attention to photons from specified source and background areas is a necessary part of astrophysical X-ray analysis. The IRAF/PROS X-ray analysis system provides the required masking capability through its regions subsystem, described by Mandel et al. (1993). The interface to the PROS regions subsystem is a language for constructing ASCII region descriptors. The language is based on a set of simple geometric shapes and a scheme for specifying complex shapes as combinations of the simple shapes. The regions subsystem supports a variety of coordinate systems, both celestial and pixel-based.

Anticipating two kinds of new requirements, we reconsidered the design of the PROS region descriptor processing subsystem. One class of new requirements would be for products other than spatial masks from region descriptors. The other new requirement would be to allow new tasks, within or outside of IRAF, to use the subsystem to generate masks or other products.

2. Functions of a Region Descriptor

The primary function of a PROS region descriptor is to create an IRAF PLIO pixel mask to select photons for analysis. Certain secondary functions were already present in PROS and SAOimage:

- to display the regions on an image;
- to document the regions used in an analysis;

- to extract specific regions and their parameters for special processing, such as validity checks;
- to transform coordinates into a different system (e.g., to move data between SAOimage and PROS, when using two different pixel-based systems).

The presence of these secondary functions in PROS, however, did not satisfy our goals of functional extensibility and open interfacing.

3. The Problems We Faced

Each of the secondary functions of region descriptor processing was implemented *ad hoc*, each in its own program with its own lexical analyzer and parser. Each parser defined the region descriptor language somewhat differently. Each parser had side effects, making it difficult to add or change functionality. There was no provision for defining new functions or novel combinations of existing functions, and the system was available only within IRAF.

4. Our Solution

We defined a set of requirements for a new implementation of the regions subsystem: (1) there should be a single language definition, and a single parser, (2) there should be a single entry to the descriptor-processing subsystem, (3) selection of functions and return of resultants should be through a simple, extensible control structure, (4) functions should be independent, so that any combination would be selectable in a single call to the parser, and (5) the interfacing of functions to the general language processing scheme should be modular, so that we could define recipes for adding and changing functionality.

We determined that the underlying metaphor of a *software CPU*, executing instructions compiled by the parser, remained a sound conceptual framework for the syntactic and semantic analysis of region descriptors. In this framework, we treat a descriptor like a program for a CPU that we emulate in software:

- *Lexical analysis* provides tokens to a compiler.
- *Compilation* produces an object program of software CPU instructions.
- *Execution* of the object program creates a mask or other resultant.

The implementation, however, needed a new layer of design to provide a general method of plugging in new functions; the program interface needed a uniform access protocol, so that we might open the subsystem to general tasks.

We retained xyacc as the language definition language. Using xyacc simplified generating a correct and reliable parser. The integral processing, with xyacc, from lexical analysis through compilation challenged flexibility that we wanted in using the lexical analysis and object program. We defined a uniform and extensible system of control flags to achieve the desired flexibility in using information at any stage of the linguistic analysis.

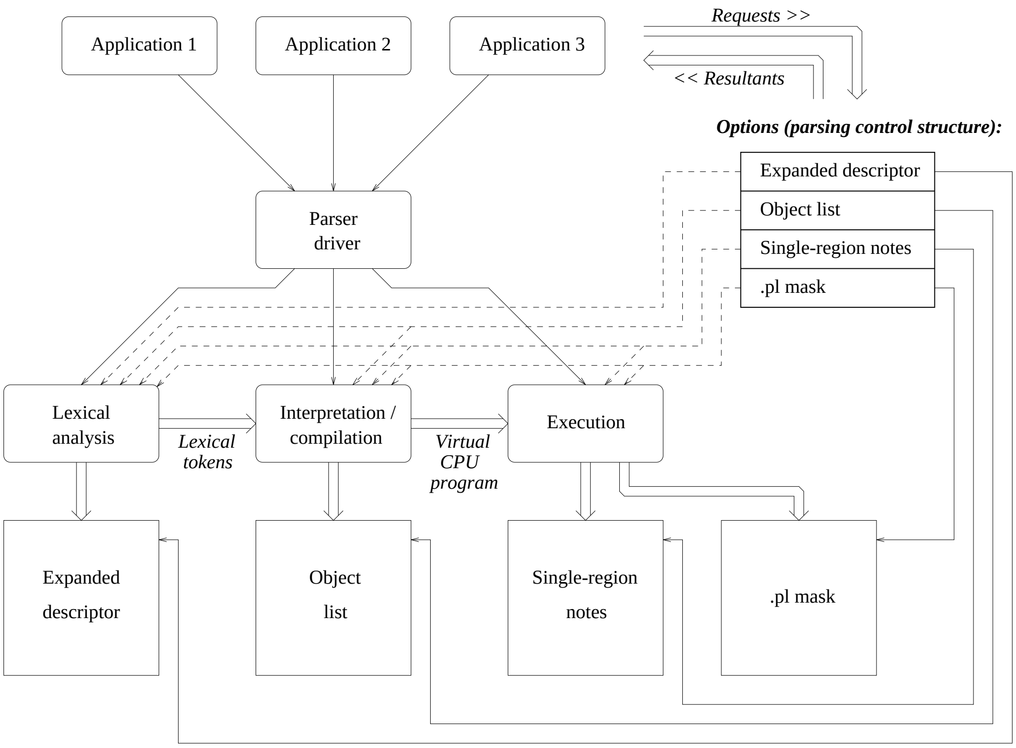

Figure 1. The PROS regions subsystem program interface.

Figure 1 shows schematically how various applications use a uniform protocol to interact with the subsystem via a versatile *parsing control structure*, and how the underlying subsystem is structured to enable functions to access, as needed, any of the three stages of the linguistic processing.

5. Using the New Design

This design prepared us for extending the regions subsystem's functionality and for making the subsystem available for use by programs in a more open environment. To make it easy for programmers to begin to exploit the new design, we created "recipes" for two ways of extending region descriptor functionality. Outlines of those procedures follow.

First, a recipe for processing a region descriptor: (1) call rg_open_parser(), to create a parsing control structure, (2) call one or more of the request setup routines, to create request/resultant structures and link them into the parsing control structure, (3) call the parser, rg_parse(), (4) retrieve the resultants through a set of macros provided, and (5) call rg_close_parser(), to release any request/resultant structures and the parsing control structure.

Second, a recipe for adding a new function: (1) define a request/resultant structure for the new option, and standard-format macros to access its fields, (2) create a slot for the new option in the parsing control structure, (3) create option setup and release routines, using standard templates, (4) integrate the option into the system of "request set query" routines and control flags, according to the processing required for the option, and (5) add the new functionality at one of four locations in the system, according to what form of the descriptor it uses. Full utilization of the new capabilities remains for future applications.

6. Conclusion

The redesign and re-implementation of the program interface to the IRAF/PROS regions subsystem has provided PROS programmers with a cleaner environment for developing new regions applications. It has been used successfully for more than a year. We anticipate greater benefit in the future, when applications outside of IRAF may make use of the new request/resultant protocol for much freer access to the functions of region descriptor processing.

Acknowledgments. This work was supported under NASA contract to the *ROSAT* Science Data Center (NAS5–30934).

References

Mandel, E., Roll, J., Schmidt, D., VanHilst, M., & Burg, R. 1993, in Astronomical Data Analysis Software and Systems II, ASP Conf. Ser., Vol. 52, eds. R. J. Hanisch, R. J. V. Brissenden, & J. Barnes (San Francisco, ASP), p. 430