descriptive commentary tags to the remote operations module (the *rosy* program will preserve the ASN.1 when it generates an abstract syntax module).

The second method is to augment the remote operations module with the `%[ ... %]` construct. This construct allows the user to select both the name used for intermediate structures that are generated and the name used for the corresponding variable in certain cases. Consider:

```
ReadImage-INV ::= SEQUENCE {
    ...

    SEQUENCE OF %[ format_list $ head %]
        FORMAT

    ...
}
```

might be translated as:

```
struct type_IMAGE_ReadImage__INV {
    ...

    struct format_list {
        struct type_IMAGE_Format *element_IMAGE_4;

        struct format_list *next;
    } *head;

    ...
};
```

More formally, the

```
%[ structure-name $ variable-name %]
```

construct can be used to control the names used when generating intermediate structures. The construct can be used with any of these ASN.1 constructs:

- `SEQUENCE %[ ... %]`
- `SEQUENCE OF %[ ... %]`

- `SEQUENCE %[ ... %] { ... }`
- `SET %[ ... %]`
- `SET OF %[ ... %]`
- `SET %[ ... %] { ... }`
- `CHOICE %[ ... %] { ... }`

If the `$ variable-name` portion of the construct is missing, then only the structure name is taken from the construct, the name of the corresponding variable is generated in the usual fashion.

NOTE: This facility is only allowed in the use of nested constructs. It is not possible to override the names chosen for the top level constructs as this would break the inheritence of structures names from other modules. E.g.,

```
Defn ::= SEQUENCE %[ ... %] { ...
```

is not supported but

```
Defn ::= SEQUENCE {
        SEQUENCE OF %[ ... %] Thing,
        ...
```

is.

5.2.2 Augmented Abstract Syntax Module

In addition to the definition of *C* language structures, *posy* will prepare an augmented abstract syntax module for use with the *pepy*(1) program. With this augmented module, *pepy* will generate *C* code to translate *C* structures to and from their corresponding abstract syntax. As this is entirely an automatic process, the contents of the augmented abstract syntax module are relatively unimportant.

```
/* automatically generated by posy 3.1 #16 (gremlin), do not edit! */

#ifndef _module_IMAGE_defined_
#define _module_IMAGE_defined_

#include <isode/psap.h>
#include <isode/pepy/UNIV-types.h>

#define type_IMAGE_Version      PElement
#define bits_IMAGE_Version      "\020\01version-1"                    10
#define bit_IMAGE_Version_version_1     0

#define type_IMAGE_Format       PElement
#define bits_IMAGE_Format       "\020\01x-windows-bitmap"
#define bit_IMAGE_Format_x_windows_bitmap       0

#define type_IMAGE_Filename     type_UNIV_GraphicString

struct type_IMAGE_Initialize_INV {
    struct type_IMAGE_Version *proposed;                              20
};

struct type_IMAGE_Initialize_RSP {
    struct type_IMAGE_Version *negotiated;
};

struct type_IMAGE_VersionsSupported {
    struct type_IMAGE_Version *supported;
};
                                                                      30
struct type_IMAGE_ReadImage_INV {
    struct element_IMAGE_1 {
        struct choice_IMAGE_0 {
            int     offset;
#define choice_IMAGE_0_file_name        1
#define choice_IMAGE_0_mail_address     2

            union {
```

```
                struct type_IMAGE_Filename *file_name;

                struct type_IMAGE_Mail_Address *mail_address;
            }       un;
        } *choice_IMAGE_0;

        struct element_IMAGE_1 *next;
    } *element_IMAGE_1;

    struct element_IMAGE_4 {
        struct type_IMAGE_Format *element_IMAGE_5;

        struct element_IMAGE_4 *next;
    } *element_IMAGE_4;
};

struct type_IMAGE_ReadImage_RSP {
    struct choice_IMAGE_1 {
        int     offset;
#define choice_IMAGE_1_x_windows        1

        union {
            struct type_IMAGE_X_Windows_Bitmap *x_windows;
        }       un;
    } *choice_IMAGE_1;
};

struct type_IMAGE_FormatsSupported {
    struct type_IMAGE_Format *supported;
};

struct type_IMAGE_Mail_Address {
    struct type_UNIV_GraphicString *local;

    struct type_UNIV_GraphicString *domain;

    PE      options;
#define bits_IMAGE_options      "\020\01default-local\02default-host"
#define bit_IMAGE_options_default_local             0
```

```
#define bit_IMAGE_options_default_host    1
};
                                                                    80
struct type_IMAGE_X_Windows_Bitmap {
    int     width;

    int     height;

    int     hot_spot_x;

    int     hot_spot_y;

    struct qbuf *data;                                              90
};
#endif
```

Figure 5.2: Example of C Language Structures

5.3 Known Deficiences

posy uses essentially the same front-end as the *pepy*(1) program, so it has some limitations in the ASN.1 syntax it can accept. Consult Section 6.3.1 for the details.

When generating the augmented abstract syntax module for *pepy*, *posy* has a well-defined set of rules. There are, however, some aspects which are still incomplete or subject to change:

- On determining if an `OPTIONAL` element is present, for elements represented by a pointer (e.g., an `OCTET STRING` and its corresponding `struct qbuf` pointer), if the pointer is non-`NULL` (non-zero valued), then the element is considered to be present.
- For elements which are represented by a scalar and not a pointer (e.g., an `INTEGER` and its corresponding `int`), an additional element, `optionals`, is created in the structure. This is a bitmask value, with each bit indicating whether a particular element is present or not. So for optional element, a test is first required to see if this element is really present. e.g.,

```
if (element -> optionals & opt_element_member1)
    /* process element -> member1 */
```

5.4 Running POSY

Here are a few things to know when running *posy*.

5.4.1 Options

The *posy* program has a few options to modify its behavior.

The '`-a`' switch directs *posy* to augment the `#include` file it generates with commentary text.

Normally, *posy* ignores all *pepy*-style augmentations except the "verbatim" actions occuring at the very beginning and end of the module. The '`-d`' switch directs *posy* to ignore the verbatim actions as well.

The '`-f`' switch directs *posy* to generate *C* routines to deallocate the structures it defines (e.g., for type `type_MODULE_type`, a routine called `free_MODULE_type`

is defined). These are appended to the augmented module definition (as a consequence, use of the '`-f`' switch forces use of the '`-d`' switch).

The '`-h`' switch enables additional heuristics when *posy* generates a *C* language structure definition. These heuristics are used to make the `structs` more concise. For example, compare the definitions in Figure 5.2 with those in Figure 5.3 starting on page 57. Both were derived from the abstract syntax module in Figure 5.1, but the definitions in Figure 5.3 was generated using the '`-h0`' switch. The current wisdom is that use of the '`-h0`' switch is a good thing. Enabling any other option also results in enabling option '`-h0`'.

The '`-h1`' switch enables "clever" but non-unique structure naming. Use of the '`-h1`' switch is **not** recommended.

The '`-h2`' switch enables the generation of arrays rather than linked-lists whenever possible. For example, compare the definitions in Figure 5.2 with the fragment in Figure 5.4 on page 58. Both were derived from the abstract syntax module in Figure 5.1, but the fragment in Figure 5.4 was generated using the '`-h2`' switch. The current wisdom makes no pronouncement as to whether the '`-h2`' switch is a good thing: some people prefer linked-lists, others perfer arrays.

The '`-o`' switch sets the name of the output file. If this switch is not specified, the standard output is used (*posy* cannot derive the name of the output file from the input file since both should have extension `.py`).

Normally, *posy* prints the name of each type as it works. The '`-s`' switch disables this behavior.

5.4.2 Makefiles

By convention, input files to *posy* have the extension `.py`. This causes something of a problem with *make*(1) as the augmented abstract syntax module which *posy* outputs should also have an extension of `py`.

Since *posy* cannot intuit the name of the output file from the input file, rather than starting with file `module.py`, we assume we are starting with file `MODULE-asn.py`. (Hence the reason for the complexity in Section 4.4.2.) Let us arbitrarily select the file named `MODULE-types.py` for the augmented abstract syntax module produced by *posy*. Hence

```
posy -o MODULE-types.py MODULE-asn.py
```

will produce:

`MODULE-types.py:` the augmented abstract syntax module; and,

`MODULE-types.h:` the *C* language structures.

Our *make*(1) rules are:

```
MODULE-types.py:  MODULE-asn.py
                  posy $(POFLAGS) -o $@ MODULE-asn.rpy
MODULE-types.h:   MODULE-types.py
```

Usually `POFLAGS` is set to "`-f -h`".

In our particular discipline, we will generate two sets of files: one for the invoker and the other for the performer.

```
MODULE-Rtypes.c:  MODULE-types.py
          pepy -a PY_advise $(PYFLAGS) -o $@ MODULE-types.py

MODULE-Itypes.c:  MODULE-types.py
                  pepy $(PYFLAGS) -o $@ MODULE-types.py
```

```
/* automatically generated by posy 3.1 #16 (gremlin), do not edit! */

#ifndef _module_IMAGE_defined_
#define _module_IMAGE_defined_

#include <isode/psap.h>
#include <isode/pepy/UNIV-types.h>

#define type_IMAGE_Initialize_INV       type_IMAGE_Version
#define free_IMAGE_Initialize_INV       free_IMAGE_Version

#define type_IMAGE_Initialize_RSP       type_IMAGE_Version
#define free_IMAGE_Initialize_RSP       free_IMAGE_Version

#define type_IMAGE_VersionsSupported    type_IMAGE_Version
#define free_IMAGE_VersionsSupported    free_IMAGE_Version

#define type_IMAGE_Version      PElement
#define bits_IMAGE_Version      "\020\01version-1"
#define bit_IMAGE_Version_version_1     0
#define free_IMAGE_Version      pe_free

#define type_IMAGE_ReadImage_RSP    type_IMAGE_X_Windows_Bitmap
#define free_IMAGE_ReadImage_RSP    free_IMAGE_X_Windows_Bitmap

#define type_IMAGE_FormatsSupported    type_IMAGE_Format
#define free_IMAGE_FormatsSupported    free_IMAGE_Format

#define type_IMAGE_Format       PElement
#define bits_IMAGE_Format       "\020\01x-windows-bitmap"
#define bit_IMAGE_Format_x_windows_bitmap     0
#define free_IMAGE_Format       pe_free

#define type_IMAGE_Filename     type_UNIV_GraphicString
#define free_IMAGE_Filename     free_UNIV_GraphicString

struct type_IMAGE_ReadImage_INV {
    struct element_IMAGE_1 {
```

```
            struct choice_IMAGE_0 {
                int     offset;                                         40
#define choice_IMAGE_0_file_name        1
#define choice_IMAGE_0_mail_address     2

                union {
                        struct type_IMAGE_Filename *file_name;

                        struct type_IMAGE_Mail_Address *mail_address;
                } un;
            } *choice_IMAGE_0;
                                                                        50
            struct element_IMAGE_1 *next;
    } *element_IMAGE_1;

    struct element_IMAGE_4 {
            struct type_IMAGE_Format *Format;

            struct element_IMAGE_4 *next;
    } *element_IMAGE_4;
};
                                                                        60
struct type_IMAGE_Mail_Address {
    struct type_UNIV_GraphicString *local;

    struct type_UNIV_GraphicString *domain;

    PE      options;
#define bits_IMAGE_options      "\020\01default-local\02default-host"
#define bit_IMAGE_options_default_local         0
#define bit_IMAGE_options_default_host  1
};                                                                      70

struct type_IMAGE_X_Windows_Bitmap {
    int     width;

    int     height;

    int     hot_spot_x;
```