NAME

xfa_alloc_trans, xfa_free_trans, xfa_alloc_state, xfa_free_state, xfa_free_attr, xfa_do_once, xfa_nfa2dfa, xfa_label_states, xfa_dfa_minimize, xfa_re2nfa

SYNOPSIS

#include <xfa.h>

xfa_trans_t *xfa_alloc_trans(xfa_system_t ****sys*, unsigned long const *****map***, unsigned long** *flags***, xfa_state_t *** *from***, xfa_**
void xfa_free_trans(xfa_system_t ****sys*, xfa_trans_t *****trn***);**
xfa_state_t *xfa_alloc_state(xfa_system_t ****sys*, int** *phsize***);**
void xfa_free_state(xfa_system_t ****sys*, xfa_state_t *****stt***, int** *recurse***);**
void xfa_free_attr(xfa_system_t ****sys*, xfa_attr_t *****attr***);**
int xfa_do_once(xfa_system_t ****sys*, xfa_state_t *****stt***, int (*****dproc***) (void *, xfa_system_t *, xfa_state_t *), void *****priv***);**
xfa_state_t *xfa_nfa2dfa(xfa_system_t ****sys*, xfa_state_t *****stt***);**
int xfa_label_states(xfa_system_t ****sys*, xfa_state_t *****stt***, unsigned long *****label***);**
int xfa_dfa_minimize(xfa_system_t ****sys*, xfa_state_t *****stt***);**
int xfa_re2nfa(xfa_system_t ****sys*, xfa_state_t **** *csts***, xfa_state_t **** *cste***, unsigned char const **** *ptr***);**

DESCRIPTION

The **XFA** library is a Finite Automata (FA) library to handle tasks like building (programmatically or from regular expressions), converting NFA (Non-deterministic FA) to DFA (Deterministic FA) and reducing FA graphs. The library relies only on the availability of an ANSI C compiler, and in this way can be used in almost every system (from embedded devices to servers). The **XFA** library uses the concepts of *state* and *transaction* to define an arbitrarly complex FA graph, where *states* are connected by *transactions*. The library supply both low level graph building capabilities and regular expression driven graph construction. The complexity of the graphs that are treatable by the **XFA** library is only limited by the amount of available memory.

Structures

The following structures are defined:

xfa_system_t

```
typedef struct s_xfa_system {
        void *priv;
        void *(*alloc) (void *, int);
        void (*free) (void *, void *);
        void (*error) (void *, int);
} xfa_system_t;
```

The **xfa_system_t** structure define the system interface for the **XFA** library. It is caller responsibility to fill up the members of the structure with the proper functions pointers that matches the host where the **XFA** library is running. The *priv* member of the **xfa_system_t** structure is an opaque pointer that is passed to all the system functions as first parameter. This can be used by the system function as "context" for the proper handling and re-entrancy of the system code. The *alloc(*priv,size*)* function is used to allocate memory. The function accepts an integer parameter that represent the size of the memory block that has to be allocated. The *alloc(*priv,size*)* function returns a pointer to the newly allocated memory block, or **NULL** in case of error. The *free(*priv,data*)* function is called to free memory blocks allocated using the *alloc(*priv,size*)* function. It is possible to pass **NULL** to *free(*priv,data*)*, and in that case no real free should be performed. The function *error(*priv,errno*)* is used by the **XFA** library to set an error code for a failing **XFA** operation.

xfa_state_t

```
typedef struct s_xfa_state {
        struct ll_list_head tlist;
        struct ll_list_head flist;
        xfa_ptrhash_t sh;
        unsigned long flags;
        unsigned long label;
        struct ll_list_head alist;
} xfa_state_t;
```

The **xfa_state_t** structure describe a state inside the FA. The *tlist* list is a *struct ll_list_head* structure that is used to links all the transactions that leave the current state. The *flist* list is a list is a *struct ll_list_head* structure that is used to links all the transactions that arrive to the current state. The *sh* field is a *xfa_ptrhash_t* structure that is used during the NFA to DFA conversion, to accumulate all the states reachable by EPS transactions. The *flags* field is a bitfield that lists all the flags associated with the state. The following flags are defined for the *flags* field:

XFASF_START This is the start of the FA graph

XFASF_TARGET This is the target of the FA graph

The *alist* field is a *struct ll_list_head* that links all the attributes of the state. Attributes are defined using the *xfa_attr_t* structure.

xfa_trans_t

```
typedef struct s_xfa_trans {
        struct ll_list_head tlink;
        struct ll_list_head flink;
        unsigned long *map;
        unsigned long flags;
        struct s_xfa_state *to;
        struct s_xfa_state *from;
} xfa_trans_t;
```

The **xfa_trans_t** structure defines a transaction inside the FA graph. One transaction connects two FA states, if one of the events described inside the **xfa_trans_t** (or the "empty" EPS event) becomes true. The *tlink* field is used to link the **xfa_trans_t** structure to the destination state, whereas the *flink* field is used to link the **xfa_trans_t** to the origin state. The *map* field is a 256 bits wide bitmap that lists all the events that would trigger the transaction. For example, if the byte 0x10 is received while in the origin state, the bit 0x10 of the *map* bitmap is checked and if found set, the transaction is executed. For EPS transactions (inside NFA graphs), the *map* field can be **NULL**. The *flags* field stores a bitmask of all the active flags for the transaction. The following flags are defined for the *flags* field:

XFATF_EPSTRANS This tells that the transaction is an "epmty" EPS transaction

The *to* field is a pointer (of **xfa_state_t** type) to the destination state of the transaction, whereas the *from* field is a pointer (of **xfa_state_t** type) to the origin state of the transaction.

xfa_attr_t

```
typedef struct s_xfa_attr {
        struct ll_list_head link;
```

```
        int type;
        void (*free) (xfa_system_t *, void *);
        void *adata;
} xfa_attr_t;
```

The **xfa_attr_t** structure desribe a generic attribute data that can be associated with a given state. The *link* field is used to link the attribute **xfa_attr_t** structure to the *alist* field of the **xfa_state_t** structure. The *type* field is a number that identifies that attribute type. In normal situations, different numbers will correspond to different types, whose data will be stored inside the *adata* field of the **xfa_attr_t** structure. The *free(*xsys,data*)* field is a pointer to an optional function that can be used to perform complex cleanup of the *adata* allocations.

Functions

The following functions are defined:

xfa_trans_t *xfa_alloc_trans(xfa_system_t ****sys*, unsigned long const ****map***, unsigned long** *flags***, xfa_state_t *** *from***, xfa_state_t *****to** **);**

Allocates a transaction structure. The *sys* parameter is the system interface to be used by the **xfa_alloc_trans** implementation. The *map* parameter is a pointer to **XFA_TRANSMAP_ULONGS** unsigned longs whose bits define the events that will trigger the transaction. If bit N (from 0 to 255) is set inside the *map* bitmap, the byte with numeric value N will trigger the transaction. The *flags* parameter specify the flags for the transaction. The following flags are defined for the *flags* parameter:

XFATF_EPSTRANS This tells that the transaction is an "epmty" EPS transaction. In this case the *map* parameter will be ignored.

The *from* parameter specify the starting state of the transaction, whereas the *to* parameter specify the destination state of the transaction. The **xfa_alloc_trans** function returns a pointer to the newly allocated transaction, or **NULL** if an error occurred.

void xfa_free_trans(xfa_system_t ****sys*, xfa_trans_t *****trn**);

Frees the transaction pointed by the *trn* parameter, using the *sys* system interface.

xfa_state_t *xfa_alloc_state(xfa_system_t ****sys*, int** *phsize***);**

Allocates a new state using the system interface passed in the *sys* parameter. The *phsize* parameter defines the initial size of the *sh* hash used during the NFA to DFA conversion to accumulate all the states reachable by EPS transactions. The constant **XFA_DEF_PHASH_SIZE** can be used as default, since the hash is automatically resized in any case. The **xfa_alloc_state** returns a pointer to the newly allocated state, or **NULL** in case of error.

void xfa_free_state(xfa_system_t ****sys*, xfa_state_t *****stt**, **int** *recurse***);**

Frees the state pointed by the *stt* parameter, using the system interface passed in the *sys* parameter. The **xfa_free_state** function also frees all the orphaned transactions that leaves/arrives the currently freed state. If the *recurse* parameter is specified, the **xfa_free_state** function will recurse and free all the nodes that are reachable from *stt* (recursively).

void xfa_free_attr(xfa_system_t *_sys_, xfa_attr_t *_attr_);

Frees the attribute pointed by *attr* using the system interface passed in the *sys* parameter.

int xfa_do_once(xfa_system_t *_sys_, xfa_state_t *_stt_, int (*_dproc_) (void *, xfa_system_t *, xfa_state_t *), void *_priv_);

Recurse through all the states reachable from the state *stt* and call the function *dproc* once per state in the reachable graph. The *sys* parameter is a pointer to the system interface, whereas the *priv* parameter is an opaque pointer that is passed to *dproc* during the state enumeration (first parameter). The second parameter passed to *dproc* is the system interface pointer, and the third one is a pointer to the currently reached state. The **xfa_do_once** function returns 0 if succeeded, or -1 if an error occurred.

xfa_state_t *xfa_nfa2dfa(xfa_system_t *_sys_, xfa_state_t *_stt_);

Create a DFA graph from the NFA graph whose starting state is passed in the *stt* parameter. The *sys* parameter is a pointer to the system interface. The **xfa_nfa2dfa** functionre turns a pointer to the starting state of the resulting DFA graph, or **NULL** in case of error.

int xfa_label_states(xfa_system_t *_sys_, xfa_state_t *_stt_, unsigned long *_label_);

The **xfa_label_states** function assign a unique label to each state of the graph whose starting state is passed in the *stt* parameter. The *label* parameter is a pointer to an unsigned long that will be updated for every assigned label. The starting value of *label will be the first assigned. The *sys* parameter is the system interface pointer. The function return 0 if succeded, or -1 in case of error.

int xfa_dfa_minimize(xfa_system_t *_sys_, xfa_state_t *_stt_);

The **xfa_dfa_minimize** function minimizes the DFA graph whose starting state is passed in the *stt* parameter. The *sys* parameter is the system interface pointer. The function returns 0 in case of success, or -1 if an error occurred.

int xfa_re2nfa(xfa_system_t *_sys_, xfa_state_t **_csts_, xfa_state_t **_cste_, unsigned char const **_ptr_);

Creates an NFA graph from a regular expression. the system interface is passed in the *sys* parameter. The resulting graph starting state is returned in *csts* while the resulting graph ending state is returned in *cste*. The regular expression is passed in the *ptr* parameter (the string pointed by *ptr) that, at the end, will be set to point to the next character after the end of the regular expression. A full regular expression syntax is supported by the **xfa_re2nfa** implementation. The function returns 0 in case of success, or -1 in case of error.

LICENSE

AUTHOR

Developed by Davide Libenzi <**davidel@xmailserver.org**>

AVAILABILITY

The latest version of **XFA** can be found at :

http://www.xmailserver.org/xfa-lib.html

BUGS

There are no known bugs. Bug reports and comments to Davide Libenzi <**davidel@xmailserver.org**>