Screen Scraping with Finite State Machines

Project for Programming Techniques

Mikel Maron
Evolutionary and Adaptive Systems
University of Sussex

December 27, 2003

1 Concept

"Screen Scraping" the Web has long been seen as a necessary, but fragile, method of loosely integrating diverse networked information sources. It's widely derided by developers for its essential inelegance, yet it's been essential; in fact, large companies, like Yodlee, have been built on only transforming a purely presentational format, HTML, into machine readable data. However, new formalized methods of distributing data and services, through XML, have received extensive attention lately, promising a loosely coupled, easily integrated, networked computing environment. Still, the Web moves fast, but not as fast as wished, and there are still vast amounts of data publicly accessible only in HTML. Screen Scraping still has a place in this gap.

Traditionally, Screen Scraping and integrating networked data required the labors of a programmer, possibly working in a scripting language with regular expressions, like Perl. Users may have perceived possible applications, but didn't have useable tools to make it happen. For instance, the many departments, groups, and schools of Sussex University publish news and events, yet there is no single centralized place to get a picture of happenings, just in this single university.

Finite State Machines are computationally equivalent to regular expressions, yet conceptually more approachable by less technical users. This project develops an HTML parsing engine based on FSMs. Further, the FSM is configurable by a simple XML format, and designed to output RSS. RSS is sometimes an abbreviation for Really Simple Syndication, and is the de facto standard for syndicating content on the Web. It's highly popular in the Weblog community, and an entire "ecosystem" of user-level tools have been developed for transformation and consumption of RSS feeds.

The software is designed to be as modular as possible, so that any component could be used separately in another project. A demonstration application has been developed, to convert data from the My Yahoo! Personalization service to RSS.

2 Design

The software was designed to be maximally modular. Code was written in C++, for Unix systems, and has been successfully compiled on Cygwin and Linux. External library requirements were kept to a minimum, aided by the flexible approach of FSMs.

Template classes are used several times, sometimes by design and sometimes necessity. The core class, FSM, requires the specification of type Input, which is the class of variables for transitions between states of the FSM. This is utilized to both develop MarkupFSM, for parsing XML and HTML into MarkupTokens, and for ScreenScraper, which uses MarkupTokens as transition variables.

Templates are also used in absence of a generic callback mechanism in C++. The FSM concept is augmented, with possible callbacks on each transition. In the case of FSMConf, the callbacks are for configuring transitions in another FSM, and in ScreenScraper for outputting structured data from HTML. There exist classes that enable callbacks on object instances, but these involved fairly complicated and messy templates themselves. Therefore, FSMConf and ScreenScraper are template classes, which take pointers to other objects on which certain transition callbacks are executed. This makes for an inelegant, but useable, system of messaging among C++ objects.

A minor issues with templates, is that they can not be compiled into intermediate object files. There

exists a ANSI C++ specification for keyword "export", which enables template object files, however no compiler has implemented "export", due to the fundamental difficulty of compiling with unknown types in a strongly typed language. Therefore, in this project, header files for templates include the code files implementing the class, rather than vice versa.

This implementation of transitions in FSM were first stored internally as an array of STL maps, linking Input objects with simple structures specifying a destination state and callback. This works fine with builtin types, and STL types, but requires, among other things, the coding of a hash function for user defined classes. So, it became simpler to just use an array of arrays of Input objects, along with a linear search function for transition mapping on each state. Most FSMs will have just a few transitions for each state, so this implementation is not very costly.

Initially, the project incorporated the expat XML parser. Expat is an event based parser, itself based on a state machine. So, it became clear that adding expat was unnecessary, that a parser addressing simply the needs of the project could be written. MarkupFSM is a seven state FSM, operating on character transitions. It is not fully compliant or validating of the HTML or XML standards, but it is functional and this class could easily be extended to specification.

MarkupToken is a simple model of a "piece" of XML or HTML, able to represent either a tag or character data. The equality operator is of interest, since it is not transitive for this class. The left side of a MarkupToken equality functions as a least constraint on the right side. That is, the right side could have attributes not present on the left, and equality would still hold. If necessary, additional operators could be defined, like operator¿ and operator¡, corresponding to additional transition functions.

FSMConf is a subclass of MarkupFSM that reads in an XML file representing an FSM, then configures an object for that FSM. This template class requires declaration of both the transition type, I, and the type of the Machine being configured, F. The internal state of an FSMConf object is represented in a single variable "parsePhase" rather than another FSM. With this implementation it seemed needlessly complicated to add this "wheel within wheel", though a later version would aim to incorporate this very elegant solution. Rather less elegantly, FSMConf parses out a "url" tag from the XML configuration file. It would have been cleaner to subclass FSMConf for this purpose, but this would have meant yet another class for this project.

Since FSMConf is meant to configure an FSM with any transition type (I), this class itself does not know how to parse the "i" tag, and its subtags. Here, there was a design choice to have either the transition type class or the machine class, handle this tag. Conceptually, it made sense to choose the transition type, but this would require unnecessary restrictions; any built-in type, or stl class, would need to be sub classed, to implement the "handleInput" method. Therefore, the machine being configured handles the "i" tag.

HTMLReader encapsulates a subset of libwww, the Internet client library from the W3C. This project needed a simple object to HTTP retrieve a document, after optionally logging in to an HTTP GET, Cookie-based authentication service. So, in addition to standard use of the library, it was necessary to implement a "Cookie Jar", to hold the authentication cookies for the main request.

At this point, a bug was discovered in the cookie handling of libwww. Libwww was not correctly parsing Cookie values, prematurely ending parsing when encountering a "=" in the value. This is a legal character in Cookie values, and commonly used. A bug report was submitted

(http://lists.w3.org/Archives/Public/www-lib/2003OctDec/0022.html), but no action has been taking yet by the W3C. In any case, this small bug fix must be applied to any libwww install in order for this project build to work correctly.

RSSWriter is a very limited class for, yes, writing RSS files. It implements a bare minimum of methods, as required by this project, and should be considered a place holder for a more complete class.

Finally, ScreenScraper is the keystone to all these classes. It is a template subclass of MarkupFSM, requiring an Output class to handle callbacks. It is configured by FSMConf, and gets its input from HTMLReader.

3 Developer Documentation

Here are described public interfaces for various classes developed in this project, and some guidelines on extending these classes.

3.1 FSM<Input>

Use of this template class requires a typedef specifying the type of Input. If this type is user-defined, it must implement operator==.

FSM()
FSM(int n)
There are two constructors, one with no arguments, the other with one integer argument specifying the number of states. The number of states must be specified at some point before any transitions are added.

void setNumStates(int n)
If not specified during construction, the number of states can be set with this method.

void AddTransition(int st, int n, Input I, string cb)
This method adds a FSM transition from state "st" to state "n", when presented with Input equivalent to "I". If "I" is null, then a Default Transition is set. Default Transitions are followed when no specified transitions are matched. "cb" specifies the callback for this transition. If there is no callback, pass in the empty string "".

void Step(Input I)
This method consults the transition table for the current state, and follows the matching transition for "I", if existing. Else, the default transition is followed, if existing. In either case, the callback is called, if existing.

void CallbackLookup(string cb)
This virtual function may be implemented by any subclasses of FSM. Based on "cb", it can perform some action, like calling another function, or simply performing some inline actions.

3.2 MarkupFSM

MarkupFSM()
No arguments necessary for the constructor

void setBuf(string b)
void setFile(string file)
Use either of these methods, to specify the text blob that the FSM will tokenize.

MarkupToken* nextToken()
Call this method, within a loop, to retrieve the next token parsed from the text. The pointer will be NULL when all text has been parsed.

3.3 MarkupToken

An entirely public class. In most cases MarkupTokens won't be constructed directly, but rather returned from a MarkupFSM.

string cdata
string tagname
map<string, string> attr
string type
These members of MarkupToken are all public accessible. "type" is set to either "cdata", "startElement", or "endElement". If type == "cdata", then cdata will be set. Otherwise "tagname" and "attr". "attr" is accessible by using an iterator, defined as map<string, string>::iterator.

void clear()
Clears all members of MarkupToken. Required by FSMConf.

bool operator==()
Equivalence is defined, but is not transitive. See Design notes for details.

void assembleTag()
This method will set cdata to the pre-parsed string, for startElement and endElement types.

3.4 FSMConf<I, F>

This template class is a subclass of MarkupFSM. It reads in an XML file, specifying an FSM, and configures an FSM object. Use the "set*" method of MarkupFSM, to specify the text blob. The input type "I", and type of the configured machine, "F", must be specified. The format of this XML file is specified in the User Documentation.

The class "F" must also specify a method "handleInput". Any cdata or tags within the <i> tag of the configuration data, will be handed off to this method. "handleInput" takes a pointer to an "I" type object, and a MarkupToken.

FSMConf(F* f)
The constructor takes a pointer to the machine being configured.

void configure()
Call this method to "do the business"

string getUrl()
If a url was specified in the configuration, it's returned.

3.5 HTMLReader

This class is a simple encapsulation of the W3C libwww library, with the addition of a "cookie jar", for logging into HTTP Get, Cookie based Authentication schemes.

HTMLReader(string u, string l=NULL)
HTMLReader()
The first constructor takes arguments "u", specifying the url for the request, and optionally "l", specifying the url for authentication. The second constructor requires that setURL be called subsequently.

void setURL(string u)
Sets the url for the request

void Retrieve()
The HTTP Request is made, with optional authentication, and the result is stored.

int Error()
Returns true if there was any problem with the request.

string Result()
Returns the result of the request.

3.6 RSSWriter

This is a simple encapsulation, for the construction of RSS 2.0 files, an XML format for syndicating content on the web. For full specification functionality, more methods will need to be written in the future.

RSSWriter()

void setTitle(string t)
void setLink(string t)
void setDescription(string t)
These methods are used to set the "title", "link", and "description" fields for the entire document.

void setItemTitle(string t)
void setItemDescription(string t)
void endItem()
These methods are used to set the "title" and "description" fields of a single item. setItemDescription may be called multiple times; the strings will be concatenated together. endItem finishes the current item, and makes ready for the next.

void outputFile(string o)
Sets the name of the file to which the RSS is written.

void print()
Writes to the specified file, or STDOUT if there is no file specified.

ScreenScraper<Output>
The keystone class for this project, and an example class for developers building other applications. The "Output" type must be specified, for handling callbacks. This is also a subclass of FSM, and configured by FSMConf, so all methods and requirements of those hold.

void setOutput(Output *o)
Sets a pointer to the object handling the output callbacks.

4 MyYahoo2RSS

My Yahoo! is a long running, popular, and comprehensive web personalization service. The quantity of content is staggering, yet the concept is now somewhat out of date with the rise of RSS aggregators. The RSS model removes the middle man from content distribution, making the supply of content practically limitless and not dependent on Yahoo's business model. Still, there is some high quality content not yet duplicated in the RSS world, such as Stock Quotes, Weather, Movie Listings (Sports Scores, TV Listings, Ski Report, and more and more).

This project fills that gap, by providing a web based service for extracting RSS feeds from the HTML of My Yahoo modules. At the core, is a command line utility, which takes three arguments: module, the name of the module to retrieve, and login and password, for authentication with Yahoo. myyahoo2rss.cpp constructs a ScreenScraper object, configured by the XML file corresponding to the desired module. The HTML is retrieved, after authentication. The configured object is run on this tokenized HTML, and RSS is generated. To make it web based, this utility is wrapped by a simple Perl CGI. It's accessible at http://brainoff.com/myy2rss/.

5 User Documentation

This User Documentation describes how to configure a screen scraping application with an XML file.

5.1 A description of the XML file

The root node is FSM.
Subnodes to FSM are url, numStates, and transition. There can be any number of transition nodes, but the number must match numStates.

Each node is represented by an integer, and the starting node is 0.

A transition requires an s and n subnodes. s specifies the "from" node, and n the "to" node, for the transition.

An i subnode to transition is optional. If not present, the transition is "default", and will operate if no

other transitions match for this state and given input.

i has subnode type, with value of either "startElement", "endElement", or "cdata". The "*Element" types require another subnode to i, tagname, matching an HTML tag.

"startElement" types can also specify attributes. attr is a subnode to i. An attr must contain an attrname subnode, and optionally an attrvalue subnode.

Finally, cb is an optional subnode to i. Its value corresponds to a callback in ScreenScraper. Accepted callbacks are "itemtitle", "itemdesc", "itemtitledesc", and "itemend". "itemtitle" adds the current token to the item's title, "itemdesc" to the item's description, "itemtitledesc" adds to both the title and description, and "enditem" finishes the current item and makes ready for the next.

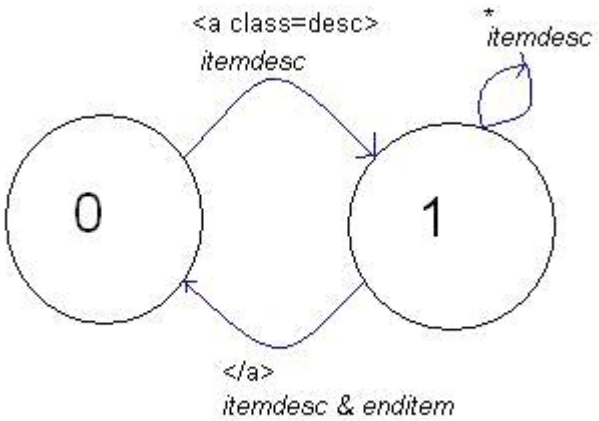
5.2 A brief example

Muxway is a simple weblog from my web acquaintance and all around friend of the web, Joshua Schacter. He doesn't supply an RSS feed for his site, so it requires some screen scraping.

Examing the HTML source for http://www.muxway.org/, items are clearly identified within the HTML, as in this example tag.

```
<b>
<a class=desc target=_new href="http://www.wellien.com/patterns/index.html">
web design patterns
</a>
</b>
```

Any time an anchor tag, with attribute class=desc, is encountered, that tag and its enclosed CDATA should be recorded in an RSS item. This requires a fairly simple FSM, with only two states and three transitions.



On encountering an anchor tag, with attribute class=desc, the FSM moves to state 1, and records the tag in the item's description. Once in state 1, the FSM stays in state 1, recording to the item description,

until the anchor end tag is seen. Then, the FSM moves back to state 0, while recording in the item's description and ending the item.
Translating to the XML file is as follows.

```
<FSM>
<url>http://www.muxway.org</url>
<numStates>2</numStates>
<transition>
        <s>0</s>
        <n>1</n>
        <i>
                <type>beginElement</type>
                <tagname>a</tagname>
                <attr>
                        <attrname>class</attrname>
                        <attrvalue>desc</attrvalue>
                </attr>
        </i>
        <cb>itemdesc</cb>
</transition>
<transition>
        <s>1</s>
        <n>1</n>
        <cb>itemdesc</cb>
</transition>
<transition>
        <s>1</s>
        <n>0</n>
        <i>
                <type>endElement</type>
                <tagname>a</tagname>
        </i>
        <cb>itemdesc,itemend</cb>
</transition>
</FSM>
```

Of course, this is a very simple example. The real power of the FSM approach comes in the more complicated FSMs needed for myyahoo2rss.

6 Future Improvements

There are a number of improvements possible to this project.

Many improvements are possible in the configuration of FSMs via XML. Since there is no XML validation, hand constructing the file can fail without much warning. Generally, more verbosity is needed during construction. At the outer limit, a GUI based configuration tool, graphically illustrating the machine, would be easiest to use and safest.

Another outer limit idea involves implementing a user level scripting language, for writing output from the screen scraper, instead of depending on routines implemented in the RSSWriter C++ class. It would

be written within the XML file. I haven't given much thought to what this would look like.

Generally, there are ideas to learn from in other screen scraping methodologies. Template::Extract is a very interesting Perl module (http://www.oreillynet.com/pub/a/javascript/excerpt/spiderhacks_chap01/). There's been success using XSL on XHTML compliant web pages (http://www.whump.com/moreLikeThis/data/21/08/2003). And the classic standard of screen scraping has been Perl & LWP (http://secu.zzu.edu.cn/Perl/Perl Bookshelf [3rd Ed]/lwp/index.htm)

Of course, there should be more My Yahoo modules available. As this only involves writing additional FSM parsers, it's an easy next step. However, the entire service requires some review, for legal implications.

It would be nice to find an implementation of real callbacks on objects, in C++, without the need for any templates. This would then allow developers to set up FSMs without subclassing.

Among the minor bugs, the intransitive equality of MarkupToken, could possibly cause some problems. For instance, when setting up transitions from a state, if transitions are equivalent, then they must be ordered from most constrained to least constrained during setup.

A Code

A.1 myyahoo2rss.cpp

```
#include "FSMConf.h"
#include "ScreenScraper.h"
#include "HTMLReader.h"
#include "MarkupFSM.h"
#include "RSSWriter.h"
#include <iostream>

using namespace std;

int main (int argc, char **argv) {

  if (argc < 4) {
    cout << "Usage: myyahoo2rss module login passwd\n";
    return 0;
  }
  string module = argv[1];
  string login = argv[2];
  string passwd = argv[3];

  typedef ScreenScraper<RSSWriter> myScraper;
  myScraper s;

  typedef FSMConf<MarkupToken, myScraper> myConf;
  myConf c(&s);
  string conffile = module + ".xml";
  c.setFile(conffile);
  c.configure();

  string url = c.getUrl();
  string loginurl = "http://login.yahoo.com/config/login?login=" + login +
"&passwd=" + passwd;
  HTMLReader h(url,loginurl);
  h.Retrieve();
  if (h.Error()) {
    return 0;
  }

  RSSWriter r;
  string out = module + "_" + login + ".rss";
  r.outputFile(out);

  r.setTitle(url);
  r.setLink(url);
  r.setDescription(url);

  s.setOutput(&r);
```

```
  MarkupFSM m;
  m.setBuf(h.Result());
  MarkupToken *t;
  t = m.nextToken();
  while (t != NULL) {
    s.Step(*t);
    t = m.nextToken();
  }

  r.print();
}
```

A.2 FSM.h

```
#ifndef FSM_READONCE
#define FSM_READONCE

/*
   Template Class for "extended" Finite State Machine
   - transition keys are of type "Input"
   - states are integers
   - a transition can be assigned a (pseudo) callback function

   Input class must implement
   - operator==
*/

#include <string>

using namespace std;

/*
  FSMValue specifies a transition for a particular state and input
*/
class FSMValue {
 public:
  int n; // new state
  bool initialized;
  string callback; // pseudo callback function
  FSMValue() { n = -1; initialized = false; callback = ""; }
};

template <class Input>
class FSM {
 private:
  int state;
  int numStates;

  /* Transitions are stored in these Lookup Tables */
  // Stores Index of next empty slot in transition tables
```

```
  int *TIndex;
  //matrix; rows are states; columns are transition keys
  Input **TransitionKeys;
  //matrix; rows are states; columns are transition values (corresponding to keys table
  FSMValue **TransitionValues;
  //table of default transitions
  FSMValue *DefaultTransitions;


  // Internal functions used by "AddTransition"
  void _DefaultTransition(int s, int n, string cb);
  void _AddTransition(int s, int n, Input i, string cb);

  // Called by constructor
  void init(int n);

 protected:
  //
  // This function should call actual functions, in response to the callback string
  // Should be overriden by child classes, to implemented "extended" fsm
  //
  virtual void CallbackLookup(string cb);

  Input curI;

 public:
  // numStates is set to n; can construct w/o it, but must setNumStates later
  FSM(int n);
  FSM() {};
  ~FSM();

  void setNumStates(int n);
  void AddTransition(int st, int n, Input i, string cb);
  void Step(Input i);
  int State() { return state; };
};

//
//With no export keyword in gcc, entire template class is defined in header
//
#include "FSM.cpp"

#endif //FSM_READONCE
```

A.3 FSM.cpp

```
//
// Details of template class FSM
// This file is included by fsm.h
//  since templates can not be compiled into object files
```

```
//

template <class Input>
FSM<Input>::FSM (int n) {
  init(n);
}

template <class Input>
void FSM<Input>::init(int n) {
  state = 0;
  numStates = n;


  TIndex = new int[numStates];
  for (int j=0; j<numStates; j++) {
    TIndex[j] = 0;
  }
  TransitionKeys = new (Input *)[numStates];
  for (int j=0; j<numStates; j++) {
    TransitionKeys[j] = new Input[10];
  }
  TransitionValues = new (FSMValue *)[numStates];
  for (int j=0; j<numStates; j++) {
    TransitionValues[j] = new FSMValue[10];
  }
  DefaultTransitions = new FSMValue[numStates];
}

template <class Input>
FSM<Input>::~FSM () {
  delete[] TIndex;
  delete[] DefaultTransitions;

  for (int j = 0; j < numStates; j++ ) {
    delete[] TransitionKeys[j];
    delete[] TransitionValues[j];
  }

  delete[] TransitionKeys;
  delete[] TransitionValues;
}

template <class Input>
void FSM<Input>::setNumStates(int n) {
  init(n);
}

template <class Input>
void FSM<Input>::_AddTransition(int s, int n, Input i, string cb) {
  if (s > numStates-1) { return; }
```

```
  if (n > numStates-1) { return; }


  TransitionKeys[s][ TIndex[s] ] = i;
  TransitionValues[s][ TIndex[s] ].initialized = true;
  TransitionValues[s][ TIndex[s] ].n = n;
  if (cb.length() > 0) {
    TransitionValues[s][ TIndex[s] ].callback = cb;
  }

  TIndex[s]++;
}

template <class Input>
void FSM<Input>::_DefaultTransition(int s, int n, string cb) {
  DefaultTransitions[s].initialized = true;
  DefaultTransitions[s].n = n;
  if (cb.length() > 0) {
    DefaultTransitions[s].callback = cb;
  }
}

template <class Input>
void FSM<Input>::Step(Input i) {
  curI = i;
  int newState;
  int oldState;

  int match = -1;
  for (int j = 0; j < TIndex[state]; j++) {
    if (TransitionValues[state][j].initialized) {
      if (TransitionKeys[state][j] == i) {
        match = j;
        break;
      }
    }
  }

  if (match >= 0) {

    oldState = state;
    newState = TransitionValues[oldState][match].n;
    if (newState > numStates -1) { //catching a fatal error
      return;
    } else {
      state = newState;
    }
    if (TransitionValues[oldState][match].callback.length() > 0) {
      CallbackLookup(TransitionValues[oldState][match].callback);
    }
```

```
  } else if (DefaultTransitions[state].initialized) {
    oldState = state;
    newState = DefaultTransitions[oldState].n;
    if (newState > numStates -1) { //catching a fatal error
      return;
    } else {
      state = newState;
    }
    if (DefaultTransitions[oldState].callback.length() > 0) {
      CallbackLookup(DefaultTransitions[oldState].callback);
    }
  }
}

template <class Input>
void FSM<Input>::CallbackLookup(string cb) {
  return;
}

template <class Input>
void FSM<Input>::AddTransition(int st, int n, Input i, string cb) {
  if (st < 0 || st >= numStates || n < 0 || n >= numStates) {
    return; //ERROR - NOTIFY?
  }

  if (! (i == (Input) NULL)) {
    _AddTransition(st,n,i,cb);
  } else {
    _DefaultTransition(st,n,cb);
  }
}
```

A.4 MarkupFSM.h

```
#ifndef MARKUPFSM_READONCE
#define MARKUPFSM_READONCE

/*
  The class implements a Finite State Machine
   for parsing markup (XML, HTML, etc)

  Given a buffer, or file, this class will iteratively return
   markupTokens

*/

#include <string>
#include <map>
#include <iostream>
```

```
#include "FSM.h"
#include "MarkupToken.h"


class MarkupFSM: public FSM<char> {
 protected:

  //
  // These members are used to keep "output state"
  //  (Yes could possibly use another FSM here, but there are some issues)
  //
  int inTag, inCdata;
  string cdata;
  string tagname;
  string attrname;
  string attrvalue;
  map<string, string> attr;

  FILE *xmlFile;
  int done;
  char buf[1024];
  char *p;

  string buffer;
  int bufi;

  MarkupToken token;
  int returnToken;

  void setupTransitions();
  void initData();


  //virtual function defined for fsm class
  void CallbackLookup(string cb);
  void tokenReady();

  MarkupToken* iterateBuffer();
  MarkupToken* iterateFile();

 public:

  MarkupFSM() : FSM<char>(8) {  setupTransitions(); initData(); xmlFile = NULL; };
  ~MarkupFSM() { }; //should close file if it's been open

  void setFile(string file);
  void setBuf(string b);
  MarkupToken* nextToken();
};
```

```
#endif
```

A.5 MarkupFSM.cpp

```
#include "MarkupFSM.h"

void MarkupFSM::setupTransitions() {
  AddTransition(0, 1, (char)NULL, "cdata");
  AddTransition(0, 2, '<', "");
  AddTransition(1, 1, (char)NULL, "cdata");
  AddTransition(1, 2, '<', "token");
  AddTransition(2, 2, (char)NULL, "tagname");
  AddTransition(2, 3, ' ', "");
  AddTransition(2, 0, '>', "token");
  AddTransition(3, 4, (char)NULL, "attrname");
  AddTransition(3, 3, ' ', "");
  AddTransition(3, 0, '>', "token");
  AddTransition(4, 4, (char)NULL, "attrname");
  AddTransition(4, 5, '=', "");
  AddTransition(4, 3, ' ', "newattr");
  AddTransition(4, 0, '>', "token");
  AddTransition(5, 7, (char)NULL, "attrvalue");
  AddTransition(5, 6, '"', "");
  AddTransition(5, 0, '>', "token");
  AddTransition(6, 6, (char)NULL, "attrvalue");
  AddTransition(6, 0, '>', "token");
  AddTransition(6, 3, '"', "newattr");
  AddTransition(6, 3, ' ', "newattr");
  AddTransition(7, 7, (char)NULL, "attrvalue");
  AddTransition(7, 3, ' ', "newattr");
  AddTransition(7, 0, '>', "token");
}

void MarkupFSM::initData() {
    cdata.clear();
    tagname.clear();
    attrname.clear();
    attrvalue.clear();
    attr.clear();
    inTag = 0; inCdata = 0;

    token.cdata.clear();
    token.tagname.clear();
    token.attr.clear();
    token.type.clear();

    returnToken = 0;
}
```

```
//for the most part, "callbacks" are entirely run here, except for tokenReady()
void MarkupFSM::CallbackLookup(string cb) {
  if (cb == "cdata") {

    inCdata = true;
    cdata += curI;

  } else if (cb == "tagname") {

    inTag = true;
    tagname += curI;

  } else if (cb == "attrname") {

    attrname += curI;

  } else if (cb == "attrvalue") {

    attrvalue += curI;

  } else if (cb == "newattr") {

    attr[attrname] = attrvalue;
    attrname = "";
    attrvalue = "";

  } else if (cb == "token") {

    if (attrname.length() > 0) {
        attr[attrname] = attrvalue;
        attrname = "";
        attrvalue = "";
    }
    tokenReady();

  }
}

void MarkupFSM::tokenReady() {

  //
  // setup the token
  //
  if (inCdata) {

    token.cdata = cdata;
    token.type = "cdata";

  } else if (inTag) {
    if (tagname[0] == '/') {
```

```
      tagname.erase(0,1);
      token.tagname = tagname;
      token.type = "endElement";
      token.cdata = "</" + tagname + ">";

    } else {

      token.tagname = tagname;
      token.attr = attr;
      token.type = "startElement";
      //token.cdata = token.reassemble();
    }
  }

  returnToken = 1;
}

void MarkupFSM::setFile(string file) {
  xmlFile = fopen(file.c_str(), "r");
  done = 0;
  p = NULL;
}

void MarkupFSM::setBuf(string b) {
  buffer = b;
  bufi = 0;
}

MarkupToken* MarkupFSM::nextToken() {
  if (xmlFile) {
    return iterateFile();
  } else if (buffer.length() > 0) {
    return iterateBuffer();
  }
}

MarkupToken* MarkupFSM::iterateBuffer() {
  initData();

  if (bufi < buffer.length()) {
    while (bufi < buffer.length()) {
      Step(buffer[bufi]);
      bufi++;
      if (returnToken) {
        return &token;
      }
    }

    tokenReady();
```

```
    return &token;
  }
  return NULL;
}

MarkupToken* MarkupFSM::iterateFile() {
  size_t len;
  initData();

  if (xmlFile && ! done) {
    do {

      while (p && *p) {
        Step(p[0]);
        p++;


        if (returnToken) {
          return &token;
        }
      }


      len = fread(buf, 1, sizeof(buf), xmlFile);
      p = buf;
      if (len < sizeof(buf)) {
        buf[len] = '\0';
      }
    } while (len > 0);

    done = 1;
    //End of File. Just return whatever we have
    tokenReady();
    return &token;
  }

  return NULL;
}
```

A.6 MarkupToken.h

```
#ifndef FSM_MARKUPTOKEN_READONCE
#define FSM_MARKUPTOKEN_READONCE

//
// Class representing a single markup (XML, HTML, etc) token
//  containing either a tag and attributes
//  or CDATA
//
```

```
#include <string>
#include <map>

using namespace std;

class MarkupToken {
 public:
  string cdata;
  string tagname;
  map<string, string> attr;
  string type;

  MarkupToken () { clear(); }
  MarkupToken(void *v) { if (! v) { clear(); } }

  //required by fsmconf
  void clear() { cdata.clear(); tagname.clear(); attr.clear(); type.clear(); };

  //
  // if type is "startElement"
  //  all attributes of "this" must match "m"
  //  but not vice-versa (m could contain elements not present in this)
  // therefore equality is not transitive!
  //

  bool operator==(MarkupToken m);

  //
  // creates a cdata field for startElements and endElements
  //
  void assembleTag();

};
#endif
```

A.7 MarkupToken.cpp

```
#include "MarkupToken.h"

bool MarkupToken::operator==(MarkupToken m) {
  if (type != m.type) {
    return false;
  }
  if (type == "cdata") {
    if (cdata.length() > 0) {
      return cdata == m.cdata;
    } else {
      return true;
    }
  }
```

```
  if (tagname != m.tagname) {
    return false;
  }

  if (attr.size() > 0) {
    map<string, string>::iterator thisIter;
    map<string, string>::iterator theirIter;

    for (thisIter = attr.begin(); thisIter != attr.end(); thisIter++) {
      theirIter = m.attr.find( thisIter->first);
      if (theirIter == m.attr.end()) {
        return false;
      }
      if (thisIter->second.length() > 0) {
        if (theirIter->second != thisIter->second) {
          return false;
        }
      }
    }
  }
  return true;

}


void MarkupToken::assembleTag() {
  if (type == "startElement") {
    cdata = "<" + tagname;
    map<string, string>::iterator iter;
    for (iter = attr.begin(); iter != attr.end(); iter++) {
      cdata.append(" " + iter->first);
      if (iter->second.length() > 0) {
        cdata.append("=\"");
        cdata.append(iter->second);
        cdata.append("\"");
      }
    }
    cdata.append(">");
  } else if (type == "endElement") {
    cdata = "</" + tagname + ">";
  }
}
```

A.8 FSMConf.h

```
#ifndef FSMCONF_READONCE
#define FSMCONF_READONCE

/*
  FSMConf: this class configures FSMs defined in an XML file
```

```
   (And just to confuse things, it uses an FSM to read that file)

  Basically an extension to MarkupFSM. Template takes I (input type)
    and F (the type of machine being configured

  Also, in a very non-generic way, will return value of the <url> field

  This class requires the Input type to have a "clear()" method

*/


#include "MarkupFSM.h"

//
// I : input type for fsm transitions
// F : the machine being configured
// R : the reader being configured
//

template <class I, class F>
class FSMConf: public MarkupFSM {
 private:
  string parsePhase;

  F *fsm;

  //
  //Keep track of the state being configured here (could possibly use FSMValue?)
  //
  int st;
  I i;
  int n;
  string cb;

  string url;

  void handleToken();
 public:
  FSMConf(F *f) { fsm = f; st = -1; n = -1; cb.clear(); i = (I) NULL; };
  ~FSMConf() { };

  void configure();
  string getUrl() { return url; };
};

//pecularity of template classes
#include "FSMConf.cpp"
#endif
```

A.9 FSMConf.cpp

```
template <class I, class F>
void FSMConf<I, F>::configure(){
  MarkupToken *t;

  t = nextToken();
  while (t != NULL) {
    handleToken();
    t = nextToken();
  }
}


template <class I, class F>
void FSMConf<I, F>::handleToken () {
  //token is member of MarkupFSM

  //
  // FSM to be configured, handles Input tags
  //
  if (parsePhase == "i") {
    if (token.type == "endElement" && token.tagname == "i") {
      parsePhase.clear();
    } else {
      fsm->handleInput(&i,token); //PROBABLY SHOULD BE PASSING AROUND OBJECT PTRS
    }
    return;
  }

  if ( token.type == "startElement") {

    if ( token.tagname == "transition" ) {
      parsePhase = "transition";
    } else if ( token.tagname == "s") {
      parsePhase = "s";
    } else if ( token.tagname == "n") {
      parsePhase = "n";
    } else if ( token.tagname == "cb") {
      parsePhase = "cb";
    } else if ( token.tagname == "i") {
      parsePhase = "i";
    } else if ( token.tagname == "url") {
      parsePhase = "url";
    } else if ( token.tagname == "numStates") {
      parsePhase = "numStates";
    }

  } else if (token.type == "endElement") {
```

```
    if ( token.tagname == "transition" ) {
      fsm->AddTransition(st,n,i,cb);
      parsePhase.clear();
      st = -1; n = -1; cb.clear(); i.clear();
    } else {
      parsePhase.clear();
    }

  } else if (token.type == "cdata") {

    if (parsePhase == "s") {
      st = atoi(token.cdata.c_str());
    } else if (parsePhase == "n") {
      n = atoi(token.cdata.c_str());
    } else if (parsePhase == "cb") {
      cb = token.cdata;
    } else if (parsePhase == "url") {
      url = token.cdata;
    } else if (parsePhase == "numStates") {
      fsm->setNumStates(atoi(token.cdata.c_str()));
    }
  }
}
```

A.10 HTMLReader.h

```
#ifndef HTMLREADER_READONCE
#define HTMLREADER_READONCE

//
// libwww headers
//
#include "WWWLib.h"
#include "WWWInit.h"
#include "WWWMIME.h"
#include "WWWHTTP.h"

#include <string>

using namespace std;

class HTMLReader {
 private:
  string url;
  string loginurl;

  HTRequest * request;
  HTChunk * chunk;
  HTAssocList * htmlreader_cookieholder;
```

```
  void SetupRequest();
  void SetupCookies();
  void Login();
  void initLibwww();

 public:
  HTMLReader(string u, string l=NULL);
  HTMLReader() { initLibwww(); };
  ~HTMLReader();

  void setURL(string u);
  void Retrieve();
  int Error();
  string Result();
};

#endif //HTMLREADER_READONCE
```

A.11 HTMLReader.cpp

```
#include "HTMLReader.h"

//
// C Callbacks for libwww
//

PRIVATE int htmlreader_terminate_handler (HTRequest * request,
HTResponse * response, void * param, int status)
{
  HTEventList_stopLoop();
  return HT_ERROR;
}


PRIVATE BOOL htmlreader_setcookie (HTRequest * request,
HTCookie * cookie, void * param)
{
  if (cookie && param) {
    HTAssocList_addObject((HTAssocList *)param, HTCookie_name(cookie),
HTCookie_value(cookie));
  }
  return YES;
}

PRIVATE HTAssocList * htmlreader_findcookie (HTRequest * request, void * param)
{
  //HTCookie deletes this list -- no need to free
  HTAssocList * temp_cookieholder = HTAssocList_new();
  HTAssoc * temp_obj;
  while (temp_obj = (HTAssoc *)HTAssocList_nextObject((HTAssocList *) param)) {
```

```
    HTAssocList_addObject(temp_cookieholder, HTAssoc_name(temp_obj),
HTAssoc_value(temp_obj));
  }
  return temp_cookieholder;
}

HTMLReader::HTMLReader(string u, string l) {
  url = u.c_str();
  if (l.length() > 0) {
    loginurl = l.c_str();
  }
  initLibwww();
}

void HTMLReader::setURL(string u) {
  url = u.c_str();
}

void HTMLReader::initLibwww() {
  //
  // Initialize libwww
  //
  HTProfile_newNoCacheClient("","");
  //HTLibInit("","");
  request = HTRequest_new();
  //HTHost_setEventTimeout(150);
  SetupRequest();
  chunk = NULL;
  HTNet_addAfter(htmlreader_terminate_handler, NULL, NULL, HT_ALL, HT_FILTER_LAST);
  HTAlert_deleteAll();

  htmlreader_cookieholder = NULL;
}

HTMLReader::~HTMLReader() {

  //
  // Clean Up libwww
  //
  if (htmlreader_cookieholder) {
    HTAssocList_delete(htmlreader_cookieholder);
    HTCookie_terminate();
  }
  HTRequest_delete(request);
  HTProfile_delete();
  //HTLibTerminate();
}

void HTMLReader::Retrieve() {
  if (loginurl.length() > 0) {
```

```
    Login();
  }
  chunk = HTLoadToChunk(url.c_str(),request);
  if (chunk) {
    HTEventList_loop(request);
  }
}

int HTMLReader::Error() {
  return ! (HTChunk_size(chunk) > 0);
}

string HTMLReader::Result() {
  string r;
  if (HTChunk_size(chunk) > 0) {
    r = HTChunk_toCString(chunk);
  }
  return r;
}

void HTMLReader::Login() {
  SetupCookies();
  chunk = HTLoadToChunk(loginurl.c_str(), request);
  if (chunk) {
    HTEventList_loop(request);
  }

  //
  // Get Ready for Second Request
  //
  HTRequest_delete(request);
  request = HTRequest_new();
  SetupRequest();

}

void HTMLReader::SetupRequest() {
  HTRequest_setRqHd(request, (HTRqHd) (DEFAULT_REQUEST_HEADERS - HT_C_USER_AGENT) );
  HTRequest_addExtraHeader(request,"User-Agent",
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.5) Gecko/20031007");
  HTRequest_setOutputFormat(request,WWW_HTML);
}

void HTMLReader::SetupCookies() {
  htmlreader_cookieholder = HTAssocList_new();
  HTCookie_init();
  HTCookie_setCookieMode((HTCookieMode) (HT_COOKIE_ACCEPT | HT_COOKIE_SEND));
  HTCookie_setCallbacks(htmlreader_setcookie, htmlreader_cookieholder,
htmlreader_findcookie, htmlreader_cookieholder);
}
```

A.12 RSSWriter.h

```
#ifndef RSSWRITER_READONCE
#define RSSWRITER_READONCE

#include <string>

using namespace std;

class RSSWriter {
 private:
  string output;
  string outFile;

  string itemtitle;
  string itemdesc;

 public:
  RSSWriter();
  ~RSSWriter() {};

  void setTitle(string t);
  void setLink(string t);
  void setDescription(string t);

  void setItemTitle(string t);
  void setItemDescription(string t);
  void endItem();

  void outputFile(string o);
  void print();
};

#endif
```

A.13 RSSWriter.cpp

```
#include "RSSWriter.h"
#include <iostream>
#include <fstream>

RSSWriter::RSSWriter () {
  output = "<?xml version=\"1.0\"?>\n<rss version=\"2.0\">\n<channel>";
}

void RSSWriter::setTitle (string t) {
  output.append("<title>" + t + "</title>\n");
}

void RSSWriter::setLink (string t) {
```

```
  output.append("<link>" + t + "</link>\n");
}

void RSSWriter::setDescription (string t) {
  output.append("<description>" + t + "</description>\n");
}

void RSSWriter::setItemTitle (string t) {
  itemtitle = t;
}

void RSSWriter::setItemDescription (string t) {
  itemdesc.append(t + "<br>\n");
}

void RSSWriter::endItem () {
  output.append("<item><title>" + itemtitle +
    "</title><description><![CDATA[" + itemdesc + "]]></description></item>\n");
  itemtitle.clear();
  itemdesc.clear();

}

void RSSWriter::outputFile(string o) {
  outFile = o;
}

void RSSWriter::print () {
  output.append("</channel></rss>");

  if (outFile.length() > 0) {
        ofstream fout(outFile.c_str());
        fout << output;
        fout.close();
  } else {
        cout << output;
  }
}
```

A.14 ScreenScraper.h

```
#ifndef FSM_SCREENSCRAPER_INCLUDE
#define FSM_SCREENSCRAPER_INCLUDE

/*
  Template Class for HTML Screen Scraper
  - class Output handles callbacks
  - implements handleInput, needed by FSMConf
*/
```

```
#include "MarkupFSM.h"
#include "FSM.h"

#include <iostream>
#include <string>

using namespace std;

template <class Output>
class ScreenScraper : public FSM<MarkupToken> {
 private:

  //Used for parsing <i>
  string parsePhase;
  string attrname;
  string attrvalue;

  Output *outputH;

 protected:

  void CallbackLookup(string cb);

 public:
  // this might belong with the Input type
  void handleInput(MarkupToken *i, MarkupToken m);

  void setOutput(Output *o);
};

#include "ScreenScraper.cpp" //template classes can't be compiled seperately
#endif
```

A.15 ScreenScraper.cpp

```
template <class Output>
void ScreenScraper<Output>::handleInput(MarkupToken *i, MarkupToken m) {
  if (m.type == "startElement") {

    parsePhase = m.tagname;

  } else if (m.type == "endElement") {

    if (m.tagname == "attr") {
      i->attr[attrname] = attrvalue;
    }
    parsePhase.clear();

  } else if (m.type == "cdata") {
```

```
    if (parsePhase == "type") {
      i->type = m.cdata;
    } else if (parsePhase == "tagname") {
      i->tagname = m.cdata;
    } else if (parsePhase == "attrname") {
      attrname = m.cdata;
    } else if (parsePhase == "attrvalue") {
      attrvalue = m.cdata;
    } else if (parsePhase == "cdata") {
      i->cdata = m.cdata;
    }

  }
}

template <class Output>
void ScreenScraper<Output>::setOutput(Output *o) {
  outputH = o;
}

template <class Output>
void ScreenScraper<Output>::CallbackLookup(string cb) {
  if (curI.type != "cdata") {
    curI.assembleTag();
  }

  if (cb == "itemtitle") {
    outputH->setItemTitle(curI.cdata);
  } else if (cb == "itemdesc") {
    outputH->setItemDescription(curI.cdata);
  } else if (cb == "itemtitledesc") {
    outputH->setItemDescription(curI.cdata);
    outputH->setItemTitle(curI.cdata);
  } else if (cb == "itemend") {
    outputH->endItem();
  }
}
```

B Support Files

B.1 Makefile

```
LIBWWW = -L/usr/local/lib -lwwwinit -lwwwapp -lwwwhtml -lwwwtelnet
-lwwwnews -lwwwhttp -lwwwmime -lwwwgopher -lwwwftp
-lwwwfile -lwwwdir -lwwwcache -lwwwstream -lwwwmux
-lwwwtrans -lwwwcore -lwwwutils -lmd5 -lz

LIBWWWINC = -I/usr/local/include -I/usr/local/include/w3c-libwww
-DHAVE_CONFIG_H
```

```
myyahoo2rss: myyahoo2rss.cpp FSM.cpp FSM.h FSMConf.cpp FSMConf.h
ScreenScraper.cpp ScreenScraper.h HTMLReader.o MarkupFSM.o
MarkupToken.o RSSWriter.o
        g++ myyahoo2rss.cpp -g MarkupFSM.o RSSWriter.o MarkupToken.o
HTMLReader.o $(LIBWWWINC) $(LIBWWW) -o myyahoo2rss

MarkupToken.o: MarkupToken.cpp MarkupToken.h
        g++ -c MarkupToken.cpp -ggdb -o MarkupToken.o

MarkupFSM.o: MarkupFSM.cpp MarkupFSM.h
        g++ -c MarkupFSM.cpp -ggdb -o MarkupFSM.o

RSSWriter.o: RSSWriter.cpp RSSWriter.h
        g++ -c RSSWriter.cpp -ggdb -o RSSWriter.o

HTMLReader.o: HTMLReader.cpp HTMLReader.h
        g++ -c HTMLReader.cpp -ggdb $(LIBWWWINC) -o HTMLReader.o
```

B.2 mymovies.xml

```
<?xml version="1.0"?>
<FSM>
<url>http://my.yahoo.com/preview/mymovies.html</url>
<numStates>12</numStates>
<transition>
        <s>0</s><n>1</n>
        <i><type>startElement</type><tagname>tr</tagname></i>
</transition>
<transition>
        <s>1</s><n>0</n>
</transition>
<transition>
        <s>1</s><n>2</n>
        <i><type>startElement</type><tagname>td</tagname></i>
</transition>
<transition>
        <s>2</s><n>0</n>
</transition>
<transition>
        <s>2</s><n>3</n>
        <i><type>startElement</type><tagname>b</tagnam></i>
</transition>
<transition>
        <s>3</s><n>0</n>
</transition>
<transition>
        <s>3</s><n>4</n>
        <i><type>startElement</type><tagname>a</tagname></i>
</transition>
<transition>
```

```
        <s>4</s><n>0</n>
</transition>
<transition>
        <s>4</s><n>5</n>
        <i><type>startElement</type><tagname>font</tagname></i>
</transition>
<transition>
        <s>5</s><n>0</n>
</transition>
<transition>
        <s>5</s><n>6</n>
        <i><type>cdata</type></i>
        <cb>itemtitle</cb>
</transition>
<transition>
        <s>6</s><n>11</n>
        <i><type>endElement</type><tagname>table</tagname></i>
        <cb>itemend</cb>
</transition>
<transition>
        <s>6</s><n>7</n>
        <i><type>startElement</type><tagname>tr</tagname></i>
</transition>
<transition>
        <s>7</s><n>8</n>
        <i><type>startElement</type><tagname>td</tagname></i>
</transition>
<transition>
        <s>7</s><n>6</n>
</transition>
<transition>
        <s>8</s><n>9</n>
        <i><type>startElement</type><tagname>small</tagname></i>
</transition>
<transition>
        <s>8</s><n>10</n>
        <i><type>startElement</type><tagname>b</tagname></i>
</transition>
<transition>
        <s>8</s><n>6</n>
</transition>
<transition>
        <s>9</s><n>9</n>
        <cb>itemdesc</cb>
</transition>
<transition>
        <s>9</s><n>6</n>
        <i><type>endElement</type><tagname>small</tagname></i>
</transition>
<transition>
```

```
        <s>10</s><n>4</n>
        <i><type>startElement</type><tagname>a</tagname></i>
        <cb>itemend</cb>
</transition>
<transition>
        <s>10</s><n>6</n>
        <i><type>startElement</type><tagname>small</tagname></i>
</transition>
</FSM>
```

B.3 quotes.xml

```
<?xml version="1.0"?>
<FSM>
<url>http://my.yahoo.com/preview/quotes.html</url>
<numStates>9</numStates>
<transition>
        <s>0</s><n>1</n>
        <i><type>startElement</type><tagname>tr</tagname>
                <attr><attrname>id</attrname></attr>
        </i>
</transition>
<transition>
        <s>1</s><n>2</n>
        <i><type>startElement</type><tagname>table</tagname></i>
</transition>
<transition>
        <s>2</s><n>0</n>
        <i><type>endElement</type><tagname>table</tagname></i>
</transition>
<transition>
        <s>2</s><n>3</n>
        <i><type>startElement</type><tagname>td</tagname></i>
</transition>
<transition>
        <s>3</s><n>4</n>
        <i><type>startElement</type><tagname>td</tagname></i>
</transition>
<transition>
        <s>4</s><n>4</n>
        <cb>itemdesc</cb>
</transition>
<transition>
        <s>4</s><n>5</n>
        <i><type>endElement</type><tagname>td</tagname></i>
</transition>
<transition>
        <s>5</s><n>6</n>
        <i><type>startElement</type><tagname>td</tagname></i>
</transition>
```

```
<transition>
        <s>6</s><n>6</n>
        <cb>itemdesc</cb>
</transition>
<transition>
        <s>6</s><n>7</n>
        <i><type>endElement</type><tagname>td</tagname></i>
</transition>
<transition>
        <s>7</s><n>8</8>
        <i><type>startElement</type><tagname>td</tagname></i>
</transition>
<transition>
        <s>8</s><n>8</n>
        <cb>itemdesc</cb>
</transition>
<transition>
        <s>8</s><n>2</n>
        <i><type>endElement</type><tagname>td</tagname></i>
        <cb>itemend</cb>
</transition>
</FSM>
```

B.4 weather.xml

```
<?xml version="1.0"?>
<FSM>
<url>http://my.yahoo.com/preview/weather.html</url>
<numStates>10</numStates>
<transition>
       <s>0</s<n>1</n>
       <i><type>startElement</type><tagname>td</tagname></i>
</transition>
<transition>
       <s>1</s><n>0</n>
</transition>
<transition>
       <s>1</s><n>2</n>
       <i><type>startElement</type><tagname>a</tagname></i>
       <cb>itemdesc</cb>
</transition>
<transition>
       <s>2</s><n>3</n>
       <i><type>cdata</type></i>
       <cb>itemtitledesc</cb>
</transition>
<transition>
       <s>3</s><n>4</n>
       <i><type>endElement</type><tagname>a</tagname></i>
       <cb>itemdesc</cb>
```

```
</transition>
<transition>
       <s>4</s><n>5</n>
       <i><type>startElement</type><tagname>td</tagname></i>
</transition>
<transition>
       <s>5</s><n>6</n>
       <i><type>startElement</type><tagname>small</tagname></i>
</transition>
<transition>
       <s>6</s><n>7</n>
       <cb>itemdesc</cb>
</transition>
<transition>
       <s>7</s><n>8</n>
       <i><type>startElement</type><tagname>td</tagname></i>
</transition>
<transition>
       <s>8</s><n>9</n>
       <cb>itemdesc</cb>
</transition>
<transition>
       <s>9</s><n>0</n>
       <cb>itemend</cb>
</transition>
</FSM>
```