ChatScript Control Scripts

© Bruce Wilcox, gowilcox@gmail.com

Revision 5-10-2015 cs5.3

Control scripts are an advanced topic you should read about ONLY after you have read the Advanced User Manual. In fact, may you never have any need to read this manual at all. What comes shipped with by default with ChatScript may well be all you ever need.

But actually, a control script is nothing more than a topic written in ChatScript, much like any other topic. It is the topic that controls how the ChatScript engine works, which means you can write bots to operate any way you choose. Want to have your bot simulate ingesting alcohol, getting drunk, and recovering over time? You can write control scripts to do that. My Suzette bot had a control script that would analyze your interactions with her. If you were friendly, agreeable, interested in the conversation, she got friendlier. Until she reached a point where she went neurotic and felt she wasn't deserving of your attention. This became clear in her conversations. Likewise if you were hostile, disagreeable, and bored with her, she drifted into paranoid. She wondered who might be listening in on the conversation, why you asked the questions you did. She even began to speculate how she might kill you. It's all in the control script she had (and topics she had that got invoked to simulate paranoia or neurotic affection).

The default control script shipped with ChatScript reacts to only the first sentence of your input. One can instead write a control script to react to each input sentence. It's a choice.

The only absolute control the engine has built in is that it will react to each input by invoking a control script before it begins looking at any sentence of your input, a control script for each sentence of your input, and a control script after all input has been processed.

Actually, the first part of controlling your bot is the outputmacro that defines what happens when your bot gets first created for a user. This is where you tell the system what topics to call for the 3 control phases and what variables you want to initialize or facts you want to create. Also what topic to start out in, if any. E.g.,

```
outputmacro: Thomas()
^addtopic(~introductions)
$cs_control_pre = ~xpre_control
$cs_control_main = ~xmain_control
$cs_control_post = ~xpost_control
$girlfriend = Mary
$cs_token = #DO_INTERJECTION_SPLITTING  | #DO_SUBSTITUTE_SYSTEM   |
#DO_NUMBER_MERGE   | #DO_PROPERNAME_MERGE   |
#DO_CONDITIONAL_POSTAG  | #DO_PARSE | #DO_SPELLCHECK
```

If you don't define $*cs*_control_pre or $*cs*_control_post, they just don't invoke anything. You need $*cs*_control_main to respond to an input. I use $*cs*_control_pre to initialize some variables before handling each sentence. I use $*cs*_control_post to have the system analyze the chatbots own output for things like pronoun resolution and current mood, and to generate special out-of-band messages to control an avatar.

The ^addtopic(~introductions) is needed if you expect the bot is going to have the first word. Typically the user "logs in" and the bot responds with a greeting message first, be it "Welcome to ChatScript" or "Welcome back" if this is not your first conversation. Since there is no user input on the login message, there are no keywords to initiate a topic. One could put the starting topic as part of the control script, but it's easy enough just to say "start here" using addtopic.

The $*cs*_token assignment tells the system how to preprocess input (parsing, pos-tagging, spelling correction, etc).

The interesting thing about the pre and post control scripts, is that they are always invoked in gambit mode (hence you use t: rules). They don't have any user input to respond to, so they can't be invoked in responder mode. The main program, as it is responding to user sentences, is always invoked in responder mode and will generally consist of u: or s: or ?: rules.

What you do in the control script is up to you. For Harry, for example, one of the things it does is call engine code for a random topic to gambit from when it has nothing better to do. But this means all topics are available to it equally. In a commercial product, the priority of topics was specified in a list. Movies, music, food are common high-priority topics. Funeral customs was a low priority topic. So we didn't want it randomly coming up early and the control script walked the list of topics in priority order trying to find a gambit from one of them.

The Harry bot main control script has a lot of ^if statement of the form:

if (%response == 0) {nofail(TOPIC ^rejoinder())}

If you provide the input "my life is fun. What is the color of the sun?" the Harry bot would only respond to my life is fun and ignore the 2nd sentence. This is because the if test shown passes only if no output has been generated yet. Once output has been generated for the first sentence, the system will not try to generate output for the second sentence. If, at the start of the control script, one wrote:

$$response = %response

and later wrote

if (%response == $$response) {...}

then it could respond to each input sentence and know when it had responded to that particular sentence instead of to just the first sentence.

Meanwhile the nofail clause is because it is conceivable you write a topic that actually fails as a topic somehow. Not normal, but possible. If it did fail, it will kill the entire rest of the control topic by failing and you'd get no output. So, as a precaution against a topic written incorrectly, the safe thing to do is wrap the call to a topic or rejoinder or whatever in a ^nofail(TOPIC …) to protect against that. It's not required, and usually not necessary. But it's a real pain when things go sour and a topic actually fails. Merely failing to match any rules is not considered a topic failure by ChatScript. You generally have to explictly fail the topic yourself in script.

The Harry control topic has a common flow of control and you should read through that.

Because control scripts are just topics and have all the richness of ChatScript available to them, you can do really weird things if you want to. One of the most unusual control scripts I've ever written was a bot to process a request for an Amazon product. It's job was to break apart the incoming product description into components that could be used to validate Amazon search results. That is, ChatScript is a tool for general manipulation of natural language, not just writing chat. For example:

> *red women's shoes not made in China*

would be decomposed into:

> *Color: red*
> *Gender: female*
> *Item: shoe*
> *NotManufactured: People's Republic of China*

and that data was passed back to a program that found products meeting that specification. So it wasn't trying to have a conversation and its control script was all about managing the product request decomposition. It recognized over a hundred different distinctions like color, flavor, texture, smell, size, value, intended age of use, price range, etc.