Help Topic: IvP Behavior Utility Functions

Spring 2020

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

IvP Behavior Utility Functions

The below describe a set of string utilities defined in the IvPBehavior superclass. Any behavior that subclasses from this class will have these functions at their disposal.

The `addInfoVars(string)` function

```
void IvPBehavior::addInfoVars(string);
```

The `addInfoVars()` function accepts a comma-separated list of MOOS variables. This is typically invoked in the behavior constructor and it allows the behavior to declare to the helm which MOOS variables the behavior will need to know about. The variables may be provided in one long comma-separated list in a single `addInfoVars()` invocation, or in multiple separate invocations, or a combination. Declaring a variable twice does no harm.

For example:

```
addInfoVars("NAV_X, NAV_Y");
addInfoVars("NAV_SPEED, NAV_HEADING");
```

The `getBufferDoubleVal(string, bool&)` function

```
double IvPBehavior::getBufferDoubleVal(string, bool&);
```

The `getBufferDoubleVal()` function returns the most recent numerical value for the given MOOS variable provided as an argument. The second argument is a reference to a Boolean argument which will be filled in with true if the variable is known to the Info Buffer, and false otherwise. If the variable is not known to the Info Buffer, the return value will be zero.

For example:

```
bool ok1, ok2;
double ownship_x = getBufferDoubleVal("NAV_X", ok1);
double ownship_y = getBufferDoubleVal("NAV_Y", ok2);
if(!ok1)
  postWMessage("NAV_X info not found in the info buffer");
if(!ok2)
  postWMessage("NAV_Y info not found in the info buffer");
```

The `getBufferStringVal(string, bool&)` function

```
double IvPBehavior::getBufferDoubleVal(string, bool&);
```

The `getBufferStringVal()` function returns the most recent string value for the given MOOS variable provided as an argument. The second argument is a reference to a Boolean argument which will be filled in with true if the variable is known to the Info Buffer, and false otherwise. If the variable is not known to the Info Buffer, the return value will be zero.

For example:

```
bool ok
string ownship_name = getBufferDoubleVal("OWNSHIP_NAME", ok);
if(!ok)
  postWMessage("OWNSHIP_NAME info not found in the info buffer");
```

The `getBufferStringVal(string)` function

```
double IvPBehavior::getBufferDoubleVal(string);
```

This function is a variation of the previous `getBufferStringVal()` function. In this case it does not take a Boolean argument. The user may simply interpret an empty string as an indication that the variable is unknown to the Info Buffer.

For example:

```
string ownship_name = getBufferDoubleVal("OWNSHIP_NAME");
if(ownship_name == "")
  postWMessage("OWNSHIP_NAME info not found in the info buffer");
```

The `getBufferCurrTime()` function

```
double IvPBehavior::getBufferCurrTime();
```

The `getBufferCurrTime()` function returns the current time according to the InfoBuffer. This is given in UTC time, the number of seconds since midnight January 1st, 1970. The InfoBuffer is updated with the current time when it the buffer is updated with incoming MOOS mail at the beginning of each helm iteration. This time stamp is then frozen as the behaviors are processed by

the helm on the present iteration. This helps preserve the idea that behavior can be thought of as running in parrallel. The order in which they are processed internally by the helm is irrelevant.

For example:

```
double my_curr_time = getBufferCurrTime();
```

The `getBufferLocalTime()` function

```
double IvPBehavior::getBufferLocalTime();
```

The `getBufferLocalTime()` function returns the current time according to the InfoBuffer. It has the same functionality as `getBufferCurrTime()` but instead returns the number of seconds since the start of the helm.

For example:

```
double my_local_time = getBufferLocalTime();
```

The `getBufferTimeVal(string)` function

```
double IvPBehavior::getBufferTimeVal(string);
```

This function returns the amount of time since the given MOOS variable has been updated in the InfoBuffer. If the helm has received mail on this variable on the present helm iteration, this function will return zero. If the helm has never received any mail for this variable, the Info Buffer knows nothing about this variable and it will return -1.

For example:

```
double info_age = getBufferTimeVal("NAV_HEADING");
if(info_age == -1)
  postWMessage("No Heading info in the Info Buffer");
if(info_age == 0)
  postMessage("HEADING_CHANGE", "true");
```

The `postMessage(string, string, string)` function

```
void IvPBehavior::postMessage(string moosvar, string value, string key="");
```

This function will result in the posting to the MOOSDB of the given MOOS variable and given string value. The optional `key` parameter may be provided to override the helm duplication filter which otherwise blocks successive postings with the same variable-value unless they have different keys. Postings are collected by the helm from each behavior and posted to the MOOSDB by the helm at the end of each helm iteration.

For example:

```
postMessage("ARRIVAL_TIME", "now");
```

The postMessage(string, double, string) function

```
void IvPBehavior::postMessage(string moosvar, double value, string key="");
```

This function will result in the posting to the MOOSDB of the given MOOS variable and given *double* value. The optional `key` parameter may be provided to override the helm duplication filter which otherwise blocks successive postings with the same variable-value unless they have different keys. Postings are collected by the helm from each behavior and posted to the MOOSDB by the helm at the end of each helm iteration.

For example:

```
postMessage("ARRIVAL_TIME", utc_time);
```

The postMessage(string, bool, string) function

```
void IvPBehavior::postMessage(string moosvar, bool value, string key="");
```

This function will result in the posting to the MOOSDB of the given MOOS variable and given *Boolean* value. The optional `key` parameter may be provided to override the helm duplication filter which otherwise blocks successive postings with the same variable-value unless they have different keys. Boolean postings are actually converted to string postings. This function is merely a convenience function. Postings are collected by the helm from each behavior and posted to the MOOSDB by the helm at the end of each helm iteration.

For example:

```
postMessage("ARRIVAL_TIME", utc_time);
```

The postRepeatableMessage(string, string) function

```
void IvPBehavior::postRepeatableMessage(string moosvar, string value);
```

This function will result in the posting to the MOOSDB of the given MOOS variable and given string value. This function is similar to postMessage(string, string, string) function, but does not make use of a key. Instead, this function guarantees that all postings are made, even if successive postings have identical variable-value pairs.

For example:

```
postRepeatableMessage("ARRIVAL_TIME", "now");
```

The `postRepeatableMessage(string, double)` function

```
void IvPBehavior::postRepeatableMessage(string moosvar, double value);
```

This function will result in the posting to the MOOSDB of the given MOOS variable and given *double* value. This function is similar to `postMessage(string, double, string)` function, but does not make use of a key. Instead, this function guarantees that all postings are made, even if successive postings have identical variable-value pairs.

For example:

```
postRepeatableMessage("ARRIVAL_TIME", 0);
```