Enabling End-User Datawarehouse Mining
Contract No. IST-1999-11993
No. TR 18-02

Using Constraints, Conditions and Assertions

Martin Scholz

Dortmund, March 12, 2002

1 An Example on how to use Constraints in the HCI

This section illustrates the use of the representation framework for the HCI. Let's assume that the case designer wants to add a step embedding the operator MISSING_VALUES_WITH_REGRESSION_SVM. In table OPERATOR_T the following information is stored[1]:

OP_ID	OP_NAME	OP_LOOP	OP_MULTI	OP_MANUAL
53	'MISSING_V._WITH_R._SVM'	'YES'	'NO'	'NO'

As stated above, this operator is LOOPABLE, not MULTISTEPABLE and not MANUAL.

The constraints for this operator can be read from the tables OP_PARAMS_T and OP_CONSTR_T:

- There is exactly one input concept *In_Con*.
- A set of BaseAttributes *PredAttr* specifies a subset of the input concept's features.
- All attributes *PredAttr* are of type scalar.
- A BaseAttribute specifies the target attribute *Target*, for which missing values shall be replaced.
- *Target* is of type scalar.
- The output is a single Concept *Out_Con* containing the predicting and target features.
- The parameters *C*, *LossFunctionPos*, *LossFunctionNeg* and *Epsilon* are numerical values. All values are strictly positive.
- The *KernelType* is one of "dot", "polynomial", "radial", "neural" and "anova".

The input/output constraints (OP_PARAMS_T) would be represented as

ID	OP_ID	MINARG	MAXARG	NAME	IO	TYPE
1018	53	1	1	"In_Con"	"IN"	"CON"
1019	53	1	NULL	"PredAttr"	"IN"	"BA"
1020	53	1	1	"Target"	"IN"	"BA"
1021	53	1	1	"C"	"IN"	"V"
1022	53	1	1	"LossFunctionPos"	"IN"	"V"
1023	53	1	1	"LossFunctionNeg"	"IN"	"V"
1024	53	1	1	"KernelType"	"IN"	"V"
1025	53	1	1	"Out_Con"	"OUT"	"CON"

[1]Omitting OP_REALIZES.

In table OP_CONSTR_T the other constraints can be stated as

CONSTR_ID	OP_ID	CONSTR_TYPE	OBJ1	OBJ2
1035	53	"IN"	"PredAttr"	"In_Con"
1036	53	"TYPE"	"PredAttr"	"SCALAR"
1037	53	"IN"	"Target"	"In_Con"
1038	53	"TYPE"	"Target"	"SCALAR"
1039	53	"TYPE"	"C"	"SCALAR"
1040	53	"TYPE"	"LossFunctionPos"	"SCALAR"
1041	53	"TYPE"	"LossFunctionNeg"	"SCALAR"
1042	53	"TYPE"	"Epsilon"	"SCALAR"
1043	53	"GT"	"C"	0
1044	53	"GT"	"LossFunctionPos"	0
1045	53	"GT"	"LossFunctionNeg"	0
1046	53	"GT"	"Epsilon"	0
1047	53	"ONE_OF"	"KernelType"	"dot,polynomial,..."
1048	53	"COPY_REPL"	"In_Con"	"Target"

The most important information for the case editor will be, which inputs and outputs belong to a certain operator. As stated in OP_PARAMS_T a step embedding the MISSING_VALUES_WITH_REGRESSION_SVM needs to be provided with an input concept, a target attribute, a set of base attributes used to predict the missing target attribute and four parameters of type VALUE. The operator produces a single output concept. Additionally to these constraints, the second table gives further restrictions. The parameters have to be of matching type and the numerical values need to be strictly positive. The user should be guided to enter the necessary information.

Together, these constraints provide the HCI with all the necessary information about the validity of steps, regarding the demands of operators. The output needs to be specified, too, when setting up a step. In this case the user does not have to fully specify the output concept by hand, but the HCI may exploit the information given by the "COPY_REPL"-constraint: The output concept will share all base attributes with the input concept, except for the target attribute. The target attribute will be a different base attribute, having the same name as the original target attribute. So the HCI could support the user by offering to create a new concept, having the specified set of features. This new concept can immediately be defined to be the output concept of the step.

After validating the set of parameters for a certain step with respect to the general constraints given for the embedded operator, all the parameters have to be entered into the table PARAMETER_T. The names that need to be specified in this table have to be the same as given in the table OP_CONSTR_T. It would be reasonable, that the case editor enters the parameters using the M4 interface.

2 When to check Conditions and how to exploit Assertions

The conditions can hardly be exploited by the case editor. To illustrate their runtime specific character, let's continue with the operator MISSING_VALUES_WITH_REGRESSION_SVM, having the following conditions:

- The predicting attributes *PredAttr* do not containing any missing values.
- the target attribute has missing values.

Represented in the table OP_COND_T:

COND_ID	OP_ID	COND_TYPE	OBJ1	OBJ2
1050	53	HAS_NULLS	"Target"	NULL
1051	53	NOT_NULL	"PredAttr"	NULL

In general it will not be possible to decide, if a base attribute contains a missing value, without looking at the data. This condition could rather be checked by the compiler at runtime, in order to avoid an unnecessary operator application, if there are no missing values, anyway. In other cases, maybe if a time attribute is not equidistant, although an equidistant time series is expected, a runtime exception with a meaningful message to the user should be generated.

The main advantage of formalizing assertions is to avoid checking conditions, which can already be concluded to hold or to be violated. For the example operator the output base attribute will not contain any missing values, which is represented in table OP_ASSERT_T as follows:

ASSERT_ID	OP_ID	ASSERT_TYPE	OBJ1	OBJ2
1035	53	NOT_NULL	"Target"	NULL

Once such an assertion is true, it can be memorized by the compiler, in order to avoid unnecessary checks at runtime.